



DEVELOPMENT OF A METHODOLOGY FOR CUSTOMIZING INSIDER THREAT
AUDITING ON A LINUX OPERATING SYSTEM

THESIS

William T. Bai
Master Sergeant, USAF

AFIT/GCO/ENG/10-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCO/ENG/10-01

DEVELOPMENT OF A METHODOLOGY FOR CUSTOMIZING INSIDER THREAT
AUDITING ON A LINUX OPERATING SYSTEM

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

William T. Bai, BS
Master Sergeant, USAF

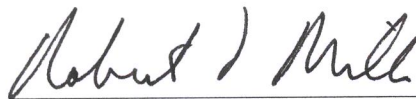
March 2010

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

DEVELOPMENT OF A METHODOLOGY FOR CUSTOMIZING INSIDER THREAT
AUDITING ON A LINUX OPERATING SYSTEM

William T. Bai, BS
Master Sergeant, USAF

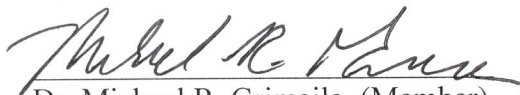
Approved:



Dr. Robert F. Mills, (Chairman)

9 MAR 10

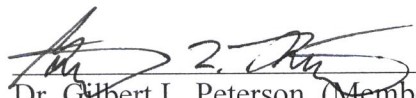
(Date)



Dr. Michael R. Grimaila, (Member)

9 MAR 10

(Date)



Dr. Gilbert L. Peterson, (Member)

9 MAR 2010

(Date)

Abstract

Insider threats can pose a great risk to organizations and by their very nature are difficult to protect against. Auditing and system logging are capabilities present in most operating systems and can be used for detecting insider activity. However, current auditing methods are typically applied in a haphazard way, if at all, and are not conducive to contributing to an effective insider threat security policy. This research develops a methodology for designing a customized auditing and logging template for a Linux operating system. An intent-based insider threat risk assessment methodology is presented to create use case scenarios tailored to address an organization's specific security needs and priorities. These organization specific use cases are verified to be detectable via the Linux auditing and logging subsystems and the results are analyzed to create an effective auditing rule set and logging configuration for the detectable use cases. Results indicate that creating a customized auditing rule set and system logging configuration to detect insider threat activity is possible.

Acknowledgments

First and foremost, I would like to give my sincere appreciation to my wife and children; they have been foremost in my thoughts throughout my endeavors. I am truly blessed to have them in my life. Their understanding in regards to the time required away from them to complete this endeavor and their positive attitudes and occasional hugs have made it easier to cope with my feelings of being overwhelmed.

I would like to thank my advisor, Dr. Robert Mills. His help with pointing me in the right direction in my research and advice on the unexpected bumps in the road were instrumental in my completing this thesis. His attitude and enthusiasm allowed me to welcome the challenge and overcome obstacles. I would also like to thank Dr. Michael Grimaldi, who has selflessly and enthusiastically contributed help, guidance and technical advice for this thesis while providing me a sounding board for perceived difficulties and avenues of research.

I am deeply indebted to my friends, fellow SNCOs, and AFIT Cyber Ops graduates, SMSgt Michael Wabiszewski and SMSgt Michael Woelfle. They not only helped me with welcoming my family and me to AFIT, but also continuously helped me along the way in advice on career, education, and the future impact of our endeavors as enlisted Cyber Ops graduates of AFIT. It's been a long road, but the end of this leg of the journey is in sight.

William T. Bai

Table of Contents

	Page
Abstract	iv
Acknowledgments.....	v
Table of Contents	vi
List of Figures	ix
List of Tables	x
I. Introduction	1
1.1 Research Motivation	1
1.2 Problem Statement	2
1.3 Assumptions.....	3
1.4 Scope.....	3
1.5 Overview	4
1.6 Thesis Organization	5
II. Background	6
2.1. Chapter Overview	6
2.2. Information Security	6
2.3. Insider Threat	7
2.3.1. Definition of Insider Threat	8
2.3.2. Insider Threat Trends	9
2.3.3. Insider Threat Characteristics	10
2.3.4. Insider Threat Attacks and Methods	11
2.3.5. Issues in Identifying Possible Insider Threats	13
2.4 Risk Management	14
2.5. Auditing	17
2.5.1. What is Auditing?	17
2.5.2. Auditing Policy	18
2.5.3. The Current Role of Auditing	20
2.5.3.1. Individual Accountability	20
2.5.3.2. Reconstruction of Events	22
2.5.3.3. Intrusion Detection.....	22
2.5.3.4. Problem Analysis	23
2.6. Related Research.....	24
2.7. Linux Logging and Auditing	25
2.7.1. Auditing Overview.....	25
2.7.2. Syslog.....	26
2.7.3. Rsyslog.....	29
2.7.4. Audit Subsystem	30
2.8. Summary	33

III. Methodology	34
3.1. Problem Definition.....	34
3.1.1. Research Goals.....	35
3.1.2. Expected Outcome	35
3.2. Approach.....	35
3.3. Extended ITFDM	36
3.4. Insider Threat Use Case Detection Verification	38
3.4.1. System Logging Configuration	39
3.4.2. Auditing Configuration	39
3.5. Data Collection	40
3.6 System Setup.....	40
3.7 Procedure	40
3.8. Use Case Scenarios	41
3.9. Evaluation Technique	42
3.10. Summary	42
IV. Use Case Verification	43
4.1. Notional Example	43
4.1.1. Example Overview.....	43
4.1.2. Example Problem Definition	44
4.2. Methodology Demonstration	44
4.2.1. Extended ITFDM use case scenarios	44
4.2.2. Use Case Scenarios	45
4.2.2.1. Gain Unauthorized System and/or Data Access	46
4.2.2.2. Conceal Malicious Behavior.....	46
4.2.2.3. Take Over and/or Corrupt Computers	47
4.2.2.4. Unauthorized File Access	47
4.2.2.5. Unauthorized File Distribution & Destruction of Critical Data..	48
4.2.3. Verification Analysis and Results.....	48
4.3. Organization Specific Auditing Policy	51
4.3. Auditing and Logging Limitations.....	51
4.4. Summary	53
V. Conclusions and Recommendations	54
5.1. Problem Summary and Research Effort	54
5.2. Conclusions of Research.....	55
5.3. Significance of Research.....	56
5.4. Future Research	56
5.5 Summary	58
Appendix A: List of Acronyms.....	59
Appendix B: NIST 800-30 Risk Assessment Methodology Flowchart.....	61
Appendix C: NIST 800-30 Risk Mitigation Methodology Flowchart.....	62
Appendix D: ITFDM: Action, Alteration, Snooping, and Elevation.....	63

Appendix E: Insider Threat Use Cases67

Bibliography97

List of Figures

Figure	Page
Figure 2.1. Breakdown of Insider Threat Cases.	12
Figure 2.2. Overlap of Insider Threat Cases.	13
Figure 2.3. Taxonomy of Observables.	17
Figure 2.4. Spiral Model Flowchart.	21
Figure 2.5. Targeted Monitoring.	22
Figure 2.6. Sample Syslog Output.	27
Figure 2.7. Sample Audit Log Entry.	31
Figure 2.8. Linux Audit System Components.	32
Figure 3.1. Root and 1st Tier ITFDM Action Leaf Nodes.	37
Figure 3.2. ITFDM Malicious Intentions.	37
Figure 5.1. Audit System Data Flow.	58
Figure B.1. Risk Assessment Methodology Flowchart.	61
Figure C.1. Risk Mitigation Methodology Flowchart.	62
Figure D.1. Insider Threat Decomposed Alteration Model.	63
Figure D.2. Insider Threat Distribution Alteration Model.	64
Figure D.3. Insider Threat Distribution Snooping Model.	65
Figure D.4. Insider Threat Distribution Elevation Model.	66

List of Tables

Table	Page
Table 2.1. Insider Definitions.	9
Table 2.2. Risk Mitigation Strategies.	16
Table 2.3. Syslog Facility and Severity Levels.....	28
Table 4.1. Insider Threat Use Cases.	49
Table 4.1. Notional Example Logging and Auditing Policy for Insider Threat.	52

A DEVELOPMENT OF A METHODOLOGY FOR CUSTOMIZING INSIDER THREAT AUDITING ON A LINUX OPERATING SYSTEM

I. Introduction

In today's world, computers are an integral part of everyday life. Their integrity and security becomes increasingly important as people come to rely on information technology. As such, computer security has become a major field of concern both in the commercial and government sectors. While external attacks have resulted in large amounts of research since the early 90's, in comparison, relatively little research has been devoted to insider threat detection.

1.1 Research Motivation

Despite the disparity in research, the more severe security incidents can be attributed to insider threat [2]. As with any cyber attack, these threats may be motivated by a number of factors ranging from greed to disgruntlement. The significant difference between an external attacker and an insider is the insider is often behind the extensive security infrastructure that ensures our computer system's integrity. Because insiders already know the system, their ability to wreak havoc, theft, or any other malicious behavior is more easily achieved than by an outside attacker.

Although insider threats are not new, they are of significant concern. The value of enterprise data on a worldwide scale has provided a tantalizing incentive to abuse privileges given to trusted workers. For example, in the United Kingdom, personal

records pilfered from databases are worth approximately £4-8 (\$8-13, USD) per record [1]. Operation Firewall, an investigations conducted by the US Secret Service, resulted in the arrest of a group trafficking in at least 1.7 million credit card numbers and other personal data worth £2 billion (\$3 billion, USD) on the open market [1]. The Privacy Rights Clearinghouse reported that from January 2005 to January 2008, more than 218 million records of personal information were stolen in the United States alone [1].

While preventing insider threat is a key component of protecting an information system, they are difficult to identify and resolve by their very nature [2]. Traditional security defenses concentrate on the external threat and are unsuited to the detection and prevention of insider activity. Currently there is no complete solution since any security measures put into place, whether by policy or technology, require the user have legitimate access to the system at some level. Technologically, monitoring and detecting insider issues can be a daunting task. Current thought is to move towards monitoring rather than preventing and using concepts currently employed in intrusion detection systems [1][2].

While Linux offers an auditing subsystem to log events from the kernel and user domains, its default use depends on the distribution used. In addition, general guidelines exist that direct enabling of the auditing subsystem, such as DISA UNIX STIG V5R1 [33], but tailoring these guidelines into a specific strategy or template on a Linux system is still needed.

1.2 Problem Statement

Defending against insider attacks is and will remain challenging. For the most part, traditional computer security defenses will not suffice. It will take a combination of

things including technical means, processes, and policy. Despite the need for a comprehensive security mechanism to enforce and monitor system usage to mitigate insider threat, there is no common methodology to determine what data needs to be collected or analyzed for the technical portion of this defense. A simple and unexplored approach to collecting this data is to use system and audit logs. Since these records can contain all events on a device, they record both normal and abnormal activity and therefore can be utilized in signature-based methods to detect insider threat. A careful consideration of how a system should be audited is warranted to help defend against such attacks. Currently, there is no defined methodology for creating such a customized auditing template for Linux systems. Therefore, the purpose of this research is to develop a methodology to create a comprehensive auditing policy to detect insider threat on a Linux operating system.

1.3 Assumptions

In order to develop an auditing strategy, an organization must have identified its security requirements to ensure a properly defined strategy or template. This includes understanding the organization's critical information and potential threats to that information. The risk assessment this entails needs to also include the potential impact of an information breach. Based on this assessment, the impact of these vulnerabilities can be prioritized to drive the security requirements in this study.

1.4 Scope

While there are a plethora of threats to a computer system from both the outside and inside, the focus of this study is on malicious insider threat because of its inherent difficulty in detecting and mitigating insider threat risks. It is proposed that a properly

configured auditing system can be effective in detecting such threats. Due to the large number of Linux distributions available, this research focuses on the analysis of a local workstation running Ubuntu v9.10, a popular desktop Linux variation.

1.5 Overview

One must first identify the threats and understand an insider's intentions and actions before determining the viability of auditing as an insider threat countermeasure. King [3][4] developed a process to identify such threats in an organization by extending the Insider Threat Functional Decomposition Model (ITFDM). Once the threats and vulnerabilities to an organization have been recognized, a process to develop an effective auditing policy must be developed in a rational and methodical way. Levoy's [5] work studied the efficient auditing settings on a Windows operating system to create a customized security template to detect insider threat.

This research applies the same concepts to the Linux operating system while using King's research to develop use cases or risks. By utilizing King's methodology, an organization can identify possible insider threat risks and test the ability of Linux auditing to detect such threats as posited by Levoy [5]. By performing content analysis of current insider threat activities and risk analysis, use case scenarios specific to the organization can be developed. These use cases can then be tested to determine whether such insider attacks are detectable using the Linux logging and auditing capabilities. By analyzing the results of such tests a customizable auditing and logging template customized to an organization's specific vulnerabilities and needs may be created. Combining the efforts of these prior studies an organization can develop a more rigorous and methodical process to create insider threat auditing templates.

1.6 Thesis Organization

This chapter has served as a brief introduction to the proposed problem and the reasoning and motivation for this line of research. Chapter 2 presents an overview of insider threat and background information pertaining to this research. Chapter 3 describes the methodology, experiment setup, and calculations used in this research. Chapter 4 describes a notional example to demonstrate the proposed methodology. Chapter 5 draws some conclusions from this research and recommendations for additional research in the area of auditing templates for insider threat.

II. Background

2.1. Chapter Overview

This chapter provides the necessary background information and concepts germane to this research. It presents the current thinking and trends of insider threat research and discusses current audit log usage in monitoring workstation activity in an organization. The chapter first discusses the information security in general and leads into a definition of insider threat. Current avenues of insider threat detection and mitigation will be covered and then lead into a discussion of system auditing and logging and its current uses. The chapter finishes with how auditing and logging may assist in the detection and mitigation of insider threat.

2.2. Information Security

Information security is a general term that covers a wide area of information processing, while maintaining usability. Organizations today depend on computer systems to conduct business and access information every day. In some organizations, the availability and integrity of data can be the difference between success and failure.

Information security has evolved over the years and the popularity of the Internet was one of the driving factors that intensified efforts in information security. An ever-growing number of organizations use computers to access resources that the Internet has to offer. From information access to electronic mail, the Internet could be thought of as one of the more important developments of the 20th century.

Due to the legacy of today's Internet based on the trust-based communication of its predecessor ARPANET, the Internet Protocol was not designed with security in mind.

The TCP/IP communications stack is open to malicious users and processes across the network due to a lack of security standards built into the protocol [6]. While developments have been made to secure Internet communications, incidents still occur to remind people that security is still an important issue.

Most industries rely on regulations and rules that are set by agreed upon standards and governed by cooperative committees or bodies. The same is true for information security. Many security consultants, vendors and the US government agree upon the standard security model known as CIA, or Confidentiality, Integrity, and Availability [7]. This model is an accepted component to assessing information security risks and establishing security policy.

The National Institute of Standards, via the Federal Information Processing Standards Publication (FIPS PUB) 199, succinctly describes the goals of these three objectives. The goal of confidentiality is to prevent or minimize unauthorized access to and disclosure of data or information [7][8]. Integrity pertains to the correctness of the data or ensuring the data is free from unauthorized or improper manipulation or destruction [7][8]. Finally availability refers to the ability to access data in a timely and reliable manner for authorized use [7][8].

2.3. Insider Threat

While insider threats pose a risk to both commercial and government organizations, the impact of such breaches in security on the military or government side could be more damaging due to possible impacts in national security and foreign relations. A prime example is the CIA espionage case involving Robert Hanssen, in which he sold secret information to the USSR for over 20 years, made over 1.4 million

dollars, and compromised at least two intelligence sources resulting in their execution [9]. In response to this high impact-low probability problem, security professionals need to proactively seek ways to combat these attacks. The issues concerning this dilemma are the fact that the perpetrators are trusted members of the organization and determining the difference between premeditated or deliberate infractions and non-malicious or accidental incidents.

2.3.1. Definition of Insider Threat

Given such an example, the intuitive notion of what an insider and insider threat means is obvious. But a more detailed meaning that provides scope and definition is required. Table 2.1 [13] gives a variety of insider definitions proposed and used by various entities within the US Government. Several different sources have defined these terms and all seem to have a common factor of trust and violation of that trust [10][11][12][13].

Despite the variations on a common theme, Matt Bishop's definition of insider threat seems to be the most succinct. An insider is "a trusted entity that is given the power to violate one or more rules in a given security policy" and, "the insider threat occurs when a trusted entity abuses that power" [10].

This definition also differentiates between the honest mistake of an authorized user and the actions of a malicious user by the use of the keyword of "abuses." This important distinction follows that authorized users may make unintentional mistakes that can typically be mitigated by properly defined security policies and education. The malicious user, however, requires some sort of security mechanism for detection, whether it is technological and/or procedural in nature [2].

Table 2.1. Insider Definitions.

Definition	Source
A knowledgeable and trusted individual who has been granted access to classified information or sensitive facilities.	DOE. Annual Report to the President.1987.
A person under the statutory, regulatory, or contractual authority of the Department of Defense, or any other person who has been granted access to DoD resources, which include but are not limited to systems, networks, information, facilities, and operations.	DoD. <i>DoD Insider Threat Implementation Plan</i> , 27 September 2007.
[A person] trusted with legitimate access rights to enterprise information systems and networks. Such trusted individuals can pose a significant threat to the enterprise and beyond.	President's National Strategy to Secure Cyberspace, February 2003.
A person who is allowed inside the security perimeter of a system and consequently has some privileges not granted outsiders.	"Cyber-Security and the Insider Threat to Classified Information," Computer Science and Telecommunications Board, National Research Council, 2000.
A person who is regarded, falsely, as loyally working for or on behalf of the organization, or who inadvertently commits security breaches.	Department of Homeland Security (DHS). Broad Area Announcement 07-09, "Cyber Security Research and Development." 17 May 2007.
Anyone with access, privilege, or knowledge of information systems and services.	The MITRE Corporation. "Understanding the Insider Threat." Advanced Research and Development Activity (ARDA) Workshop Presentation, March 2004.
Anyone who is or has been authorized access to a DoD information system, whether a military member, a DoD civilian employee, or employee of another Federal agency or the private sector.	DoD. "DoD Insider Threat Mitigation: Final Report of the Insider Threat Integrated Process Team." 24 April 2000.
Individuals who were, or previously had been, authorized to use the information systems they eventually employed to perpetrate harm.	USSS and CMU SEI CERT/CC. "Insider Threat Study: Illicit Cyber Activity in Banking and Finance Sector." August 2004.
Range of definition includes incompetent users making critical mistakes to moles who have been recruited, trained, and planted by nefarious outsiders	National Research Council Computer Science and Telecommunications Board. "Cyber-Security and the Insider Threat to Classified Information." 2000.
Assigned personnel (military or civilian), host-country nationals (military or civilian), third country nationals (contract employees) or other persons assigned to or transiting the area of interest	US Air Force. <i>Force Protection</i> . Air Force Doctrine Document 2-4.1, 29 October 1999.

2.3.2. Insider Threat Trends

Recent studies of cybercrime summarize the threat insider present to both the commercial and government sector. In the 2007 E-Crime Watch Survey conducted by CSO Magazine with Carnegie Mellon's CERT, insiders were the cause of 31% of the reported incidents in which a perpetrator could be identified and 49% of all respondents reported at least one malicious insider event [14]. Although the proportion of insider

threat events has remained proportionately stable (27% in 2006), the IT security budgets have decreased by 5% and corporate security spending has been reduced by 15% [14] [15]. The 2009 CSI Computer Crime and Security Survey showed that 43.2% of respondents reported a financial loss of some sort due to malicious insiders, while financial loss itself increased 15% from 2005 [16].

The reported losses do not necessarily comprise the total damage caused by insiders. Naturally the loss of reputation or negative publicity can also cause intangible losses that may reflect in future gains or losses by an organization. In fact, such perceptions may produce reluctance to voluntarily release negative information. Such reasoning is why CERT believes that insider events may be under reported in the E-Crime Watch Surveys [17]. Such concerns have been related to the top levels of government in an ever growing concern with insider threat [18].

2.3.3. Insider Threat Characteristics

Detection and mitigation of insider threat is a difficult challenge, but the potential benefit in avoiding a loss of major consequence justifies further research and development. For example, an insider stealing blueprints containing trade secrets worth \$100 million, and selling them to a competitor to obtain a new job demonstrates the possible impact insider threat poses [11].

Insider attacks may be motivated by a range of factors, such as greed, revenge, espionage, and such issues or problems may also be exacerbated by factors outside the workplace. Coupling these factors with the fact that insiders are already within the organization and have some authorized use of the IT systems makes them well-placed to misuse a system if the inclination occurs [2]. The insider's knowledge of the system(s)

involved puts them in a better position to know what is of value, know system vulnerabilities, and hide evidence of their activities. Therefore, their ability to commit and conceal malicious activities is often better than that of outsiders.

The insider threat study released in 2008 by Carnegie Mellon discovered that insiders did not share common demographic characteristics and the attacks required limited technical characteristics [17]. The insider threat studies conducted by Carnegie Mellon's CERT and the US Secret Services NTAC into various industry sectors shows a wide range of technical abilities needed in the various attack cases. For example in the information technology and telecommunication sector 63% of the insiders held technical positions and 58% of those cases used sophisticated tools, while in the finance sector only 23% of the insiders held technical positions and 87% of the attacks required simple, legitimate user commands [17][19].

2.3.4. Insider Threat Attacks and Methods

A computer system user typically falls into one of three categories [1][2]:

- Authorized/Normal Users – individuals that use IT resources appropriately and in accordance with defined policies
- Unauthorized Malicious Users
 - Masqueraders – individuals that mask their identity to compromise the security system
 - Clandestine users – individuals that try to evade access controls and auditing measures
- Misfeasors/Authorized and Dangerous Users – individuals authorized to use the system and intentionally violate their trust to perform some malicious act

Authorized users may make unintentional mistakes, but these can typically be mitigated by properly defined security policies and education. The malicious user

requires some sort of security mechanism for detection, whether it is technological and/or procedural in nature [2].

The numerous possible attacks perpetrated by malicious insiders tend to fall into four categories: IT sabotage, theft or modification for financial gain, theft or modification for business advantage, and miscellaneous [11]. Figure 2.1 [11] shows the breakdown of 190 insider threat attacks, contained in CERT's case files, into four categories.



Figure 2.1. Breakdown of Insider Threat Cases.

One should note that some cases fell into multiple categories. For example, an insider may have committed IT sabotage and then attempted to extort money by offering to assist them in recovery effort. In such a situation the case is categorized as both IT sabotage and theft or modification of information for financial gain. Five of the 190 cases were classified in multiple categories. The overlap of categories is shown in Figure 2.2 [11].

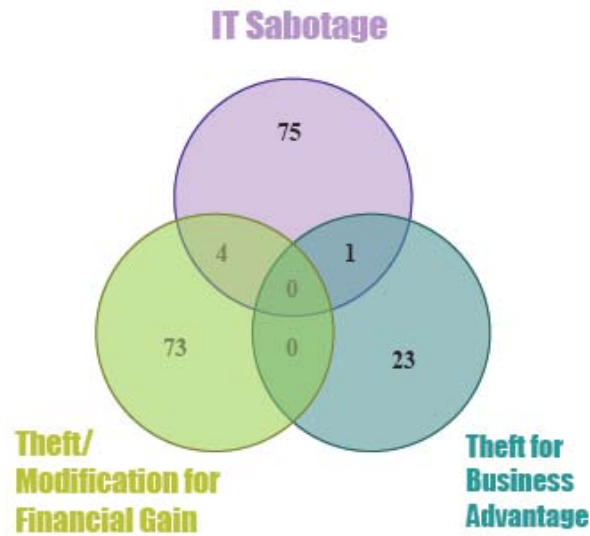


Figure 2.2. Overlap of Insider Threat Cases.

The cases of IT sabotage typically were carried out by insiders who held highly technical positions and used sophisticated technical means to carry out their attacks. Thirty percent of the insiders used their own username and password, while another 43% compromised an account. Technical means commonly used script or program writing, creating a backdoor account for later use, planting a virus, using password crackers, and installation of remote system administration tools [11].

The remaining two categories referenced in Figure 2.2, showed that the method of attack was abusing authorized permissions in a majority of cases (75% - Financial Gain, 88% - Business Advantage). And when stealing data for financial gain, 85% of the insiders used their own usernames and passwords [11].

2.3.5. Issues in Identifying Possible Insider Threats

These studies reinforce the fact that identifying possible insider attacks is difficult. A majority of the attackers used legitimate access and authorized system commands to conduct their attacks. The issues concerning this dilemma are the fact that

the perpetrators are trusted members of the organization and determining the difference between premeditated or deliberate infractions and non-malicious or accidental incidents is difficult and time consuming.

While organizations recognize that insider threat is important and must be dealt with to mitigate possible damages from attacks, the major emphasis for protection is at the procedural and policy level with rather little emphasis on possible technical means to assist in mitigating insider threat [14].

Insiders have a significant advantage in harming an organization. Insiders can bypass security measures, physical and technical, installed to prevent unauthorized access. Technological measure such as firewalls and intrusion detection systems are used to mainly defend against external threats. In addition, they have knowledge of policies, procedures, and technologies used in their organizations, and are often also knowledgeable about vulnerabilities [11].

Given the fact that most insider attack perpetrators were identified by analyzing various system logs on the system [11], it stands to reason that creating a comprehensive approach to designing a template to detect potentially harmful behavior using audit and system logs is a viable method for insider threat mitigation and detection.

2.4 Risk Management

The implementation of security controls via auditing is dependent upon an information security policy that involves the management of risk. Understanding and utilizing risk management is the foundation for an effective information security policy and provides an effective method for selecting appropriate security controls for an information system.

The DOD defines risk as the probability of loss or damage and is a function of three variables: criticality, vulnerability and threat [20]. Risk management is a process that allows organizations to balance the costs of protective measures and achieve gains in capability by protecting information systems and data that support the organization. To do this, risk management encompasses three processes: risk assessment, risk mitigation, and evaluation and assessment [21].

Risk assessment is the first process in the risk management methodology proposed by NIST. It provides a means to identify threats that pose a risk to system security. It allows one to identify the risks to system security and determining probability of occurrence, resulting impact, and additional impact mitigation measures [21]. This assessment can be accomplished quantitatively, qualitatively or via a combination of the two to prioritize and determine what critical data needs to be protected. Assessing risks to critical data enable an organization to identify and reduce those risks to an acceptable level. The NIST risk assessment methodology flowchart is provided in appendix B.

Key components of a risk assessment model are threats and vulnerabilities. It is theoretically possible to have vulnerability with no threat, but threats inherently exist for critical data residing within information systems and make vulnerability the true key to the risk assessment process.

After conducting a risk assessment, an organization would continue into the second process of risk management, risk mitigation. This process consists of prioritizing, evaluating, and implementing appropriate risk-reducing controls recommended from the risk assessment process. Although theoretically possible, the elimination of all risk is normally impractical or nigh impossible, it is the responsibility of organization managers

to use the least-cost approach and implement the most appropriate controls to decrease mission risk to an acceptable level. This is normally done by implementing one of six commonly used mitigation strategies for each vulnerability listed in Table 2.2 [21]. Once vulnerabilities are assessed, analyzing potential impacts will assist organizations in determining which risk mitigation strategy is most appropriate.

Table 2.2. Risk Mitigation Strategies.

Risk Assumption	"To accept the potential risk and continue operating the IT system or to implement controls to lower the risk to an acceptable level."
Risk Avoidance	"To avoid the risk by eliminating the risk cause and/or consequence (e.g., forgo certain functions of the system or shut down the system when risks are identified)."
Risk Transference	To transfer the risk by using other options to compensate for the loss, such as purchasing insurance."
Risk Limitation	"To limit the risk by implementing controls that minimize the adverse impact of a threats exercising vulnerability (e.g., use of supporting, preventive, detective controls)."
Risk Planning	"To manage risk by developing a risk mitigation plan that prioritizes, implements, and maintains controls."
Research and Development	"Research and Acknowledgment. To lower the risk of loss by acknowledging the vulnerability or flaw and researching controls to correct the vulnerability."

The methodology of selecting a risk mitigation strategy is outlined in appendix C.

In many organizations, periodic evaluation and assessment is most often the most overlooked process despite it being the third process in an effective risk management policy. An organization's information system will most likely continue to expand along with periodic updates, its components change, and its software applications replaced or updated with newer versions. These changes mean that new risks will surface and risks previously mitigated may again become a concern. Thus, the risk management process is ongoing and evolving and requires the third process of periodic evaluation and assessment of its risk management strategy.

2.5. Auditing

Although this research supposes that system auditing or monitoring is a viable method to detect insider threat, the basics of auditing and its capabilities must be understood. Insider threat detection is a difficult problem, but insider threat behavior can create a number of observables that can be detected by auditing. The 2004 RAND workshop developed a taxonomy of such observables (shown in Figure 2.3 [22]) to recognize insider threat characteristics. A majority of these observables were categorized as cyber actions which by their nature are auditable events on a computer system.

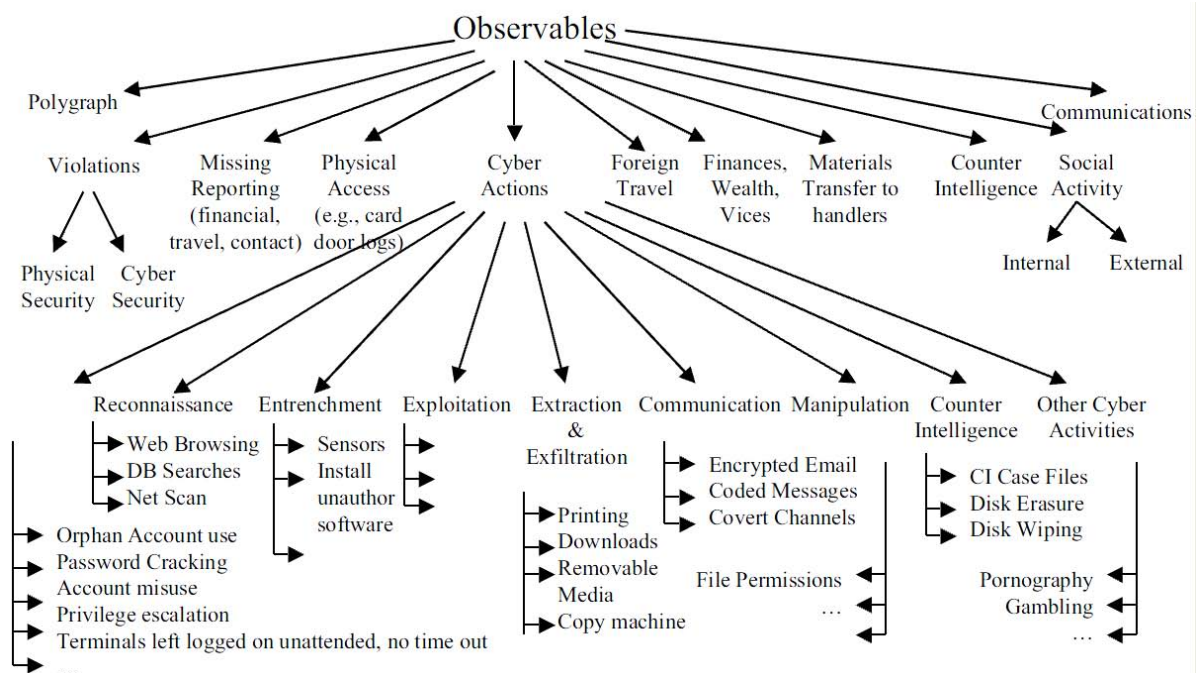


Figure 2.3. Taxonomy of Observables.

2.5.1. What is Auditing?

The National Institute of Standards and Technology defines auditing and monitoring in the following way [23]:

To maintain operational assurance, organizations use two basic methods: system audits and monitoring. These terms

are used loosely within the computer security community and often overlap. A system audit is a one-time or periodic event to evaluate security. Monitoring refers to an ongoing activity that examines either the system or the users. In general, the more "real-time" an activity is, the more it falls into the category of monitoring.

In addition, NIST also differentiates between auditing and audit trails by defining auditing as the action taken to “review and analysis of management, operational, and technical controls”, while audit trails are the “series of records of computer events, about an operating system, an application, or user activities” [24]. Needless to say audit trails are typically stored in system log files and can provide detailed information on the operation and actions taken on a computer system.

Auditing is mandated as one of the minimum security requirements for federal information and information systems to allow the “monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity” while ensuring actions of individual system users can be traced to those users for accountability [25].

The Software Engineering Institute includes auditing, logging, and monitoring as a best practice in detecting and preventing insider threats [11]. All three actions can help an organization to discover and investigate suspicious insider actions more quickly thereby lessening any possible damages. Auditing should involve the review and verification of changes to any of the organization’s critical information assets.

2.5.2. Auditing Policy

The purpose of an auditing policy is to help protect an organization's information system. Through the selection and application of well-chosen auditing rules and

procedures, auditing can help by protecting resources, reputation, legal position, and other tangible or intangible assets. To create a well-developed audit policy, an organization needs to understand how each information system supports their mission. After a system's role has been defined, the security requirements implicit in that role can be defined and be explicitly stated to define how auditing integrates into the organization's security policy.

Since computer systems are ever-changing, system security is more of an ongoing process than a static situation. Periodic reassessments of auditing policies are necessary and should ensure applicable laws and regulations are maintained. The audit policy should also describe how the system is audited, what activities should be identified, and how particular anomalies may be recognized. Also general log management functions need to be determined and outlined. For example, who will review the audit logs, how often they are reviewed, storage requirements, and incident handling.

A disconcerting trend is that many organizations fail to follow through in their policies by failing to have incident response plans. The 2007 study by Carnegie Mellon's CERT Coordination Center and CSO Magazine revealed that 48% of respondents had no formal incident handling plan, while 62% of the e-crimes were not prosecuted due to a lack of sufficient evidence or inability to identify the perpetrator. This is especially disconcerting since 74% of the insider attacks used either their own account or a compromised account [14]. Without such a plan delineating reporting and response procedures, an organization may have difficulty in determining if a security breach has even occurred.

2.5.3. The Current Role of Auditing

An organization's audit policy determines the security events to record. Organizations can monitor security-related activity and these audited events are instrumental in identifying an insider event. Failing to monitor key activities can hinder the ability to identify insider threat. While auditing is a necessary control measure, capturing excessive amounts of events will fill audit logs with trivial events and hinder the filtering process. An organization must find a balance between capturing as many events as necessary to meet their security needs and usability and practical monitoring limitations of the system. In addition, monetary costs of auditing must be balanced with the perceived benefit of such security measures. Such issues as storage space, processing power, and investigations of suspicious activity can play an important factor in how much an organization wishes to invest into auditing for insider threat.

2.5.3.1. Individual Accountability

As a technical mechanism, audit trails help managers maintain individual accountability. Proper user behavior can be promoted by advising them that they are tracked by an audit policy that logs user activities and therefore personally accountable for their actions. The likelihood of users circumventing security policy decreases if they know their actions are audited. Audit trails can work with logical access controls to restrict use of system resources and even detect improper data modifications. Such data modifications could be determined by comparing actual changes of data and expected changes. This can help determine if errors were made by the user, the system/application software, or some other source. Audit trail analysis is useful in examining user actions to ensure users are not abusing their legitimate access nor gaining unauthorized access to

resources. An example would be a user that has access to personnel records and is printing records exceeding the average user; this could indicate authorized access abuse for financial gain [24].

A RAND study conducted to understand the insider threat within the US Intelligence Community developed an attack model, detailed in Figure 2.4 [22], that a malicious insider would follow. This the risk analysis step in this deliberate decision making process would allow the attacker to assess the risk of detection prior to committing the attack. With a strong audit policy in place, the insider would expect the detection risk as high and possibly decide to stop the attack. This is how auditing could serve as a deterrent to malicious insider activity.

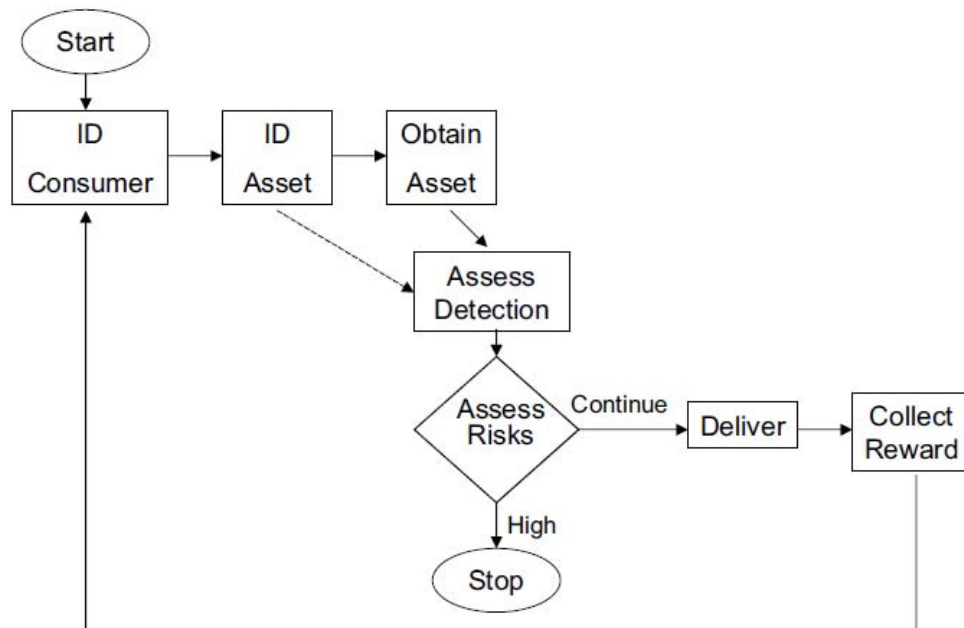


Figure 2.4. Spiral Model Flowchart.

Auditing can also be a benefit in identifying individuals more likely to perform an insider attack. A study in system dynamics on the insider threat problem described the relationship between the perceived risk of an insider, monitoring, and the discovery of

precursor activity [26]. This relationship is depicted in Figure 2.5 [26]. These precursor activities would allow an organization to pinpoint individuals at a higher risk of insider activity and implement a targeted investigation or monitoring if necessary.

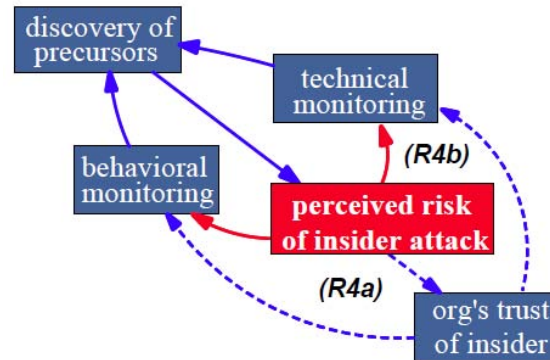


Figure 2.5. Targeted Monitoring.

2.5.3.2. Reconstruction of Events

Audit trails or logs can be used to reconstruct events. This is typically done after the detection of abnormal activity, errors, or anomalous activity, which may present threats to an organization. Such situations can be assessed by examining a systems audit trail and help determine whether an error is operator induced or system-based. Step-by-step recreation of events courtesy of an audit trail or log represents a beneficial tool for an organization's system administrators. Such records can also aid in the recovery of corrupted data files if transaction-based events are recorded [24].

2.5.3.3. Intrusion Detection

Intrusions can be detected in real or near real time, by examining audit records. Such logs can be examined as they are created or after the fact. While real-time intrusion detection is used primarily to detect outside attacks, it may also be used to detect changes in the system's performance indicative of such things as a worm or virus attack [24].

Real-time detection methods must be balanced with system usability as such processes could hamper system performance. Post-mortem identification can be used to identify and assess unauthorized access attempts and whether they were successful. Such identifications in either real-time or after the fact allows an organization to mitigate any damage occurred; the organization can also review and revise its control mechanisms.

Intrusion detection systems may also be used to automatically report attempts to compromise a system and identify and block information gathering attempts by an attacker. Such reports on attempted attacks would allow an organization to initiate incident response actions to minimize possible damage caused by the incident. The IDS may log information that could be used by the incident handlers. In addition, IDSs can be configured to report security policy violations and monitor file transfers to identify suspicious activity, such as copying large files [27]. In identifying reconnaissance activity, it is possible for an IDS to block reconnaissance attempts and notify appropriate personnel for actions, if needed, to modify security policy controls to prevent system compromises.

2.5.3.4. Problem Analysis

Audit trails may also be used in conjunction with performance monitoring tools to help identify problems. Often referred to as real-time auditing or monitoring, it may be implemented to monitor the status of critical processes and used as verification that a system is operating normally [24]. Such use could be used to detect significant increase in the use of system resources which might indicate a security problem.

2.6. Related Research

This research combines the efforts of prior studies into insider threat vulnerability assessment processes and auditing template creation, in order to develop a more rigorous and methodical process to create insider threat auditing templates. This section provides an overview into these previous studies.

King's [3] Malicious Insider Composite Vulnerability Threat Assessment methodology provides a process to identify such threats in an organization. Before determining the viability of auditing as an insider threat countermeasure, one must identify the threats and understand an insider's intentions and actions. King identified these threats by using the Cyber Observable and Attack Models [22] and extending the Insider Threat Functional Decomposition Model (ITFDM) [4]. The ITFDM study described approaches for identifying and classifying the more prevalent insider threat. King used this research and a multi-dimensional approach: content analysis, attack tree framework, and an intent driven taxonomy model to develop a malicious insider Decision Support System (DSS) tool. This study provided an assessment of an insider threat's vulnerability levels based upon aggregated vulnerability assessment and impact assessment levels [3].

This research uses King's methodology to allow an organization to identify possible insider threat risks and test the ability of Linux auditing to detect such threats as posited by Levoy [5] within a computer system. Levoy's work studied the efficient auditing settings on a Windows operating system to create a customized security template to detect insider threat using a systems inherent auditing capabilities [43]. Specifically, exploring what can be done within an organization using existing workstation auditing

tools to identify potential insider threats. His research studied what configuration of auditing on individual workstation can better identify behavior associated with malicious insider activity. This research applies the same concepts to the Linux operating system while using King's research to develop use cases or risks associated with a notional example.

2.7. Linux Logging and Auditing

In order to understand the methodology presented in this thesis, the current types of auditing being used, the auditing categories, and how they function within a Linux operating system must first be understood.

2.7.1. Auditing Overview

According to the National Institute of Standards and Technology, the most common types of security-related operating system data are system events and audit records [28].

System events are actions performed by operating system components. Many operating systems permit administrators to specify which events are logged, but failed events and the more significant successful events are the more commonly logged events. The details logged for each event can vary, but each event is usually time-stamped. Other information in the log events can include status, error codes, service name, and user or system account information for each event.

Audit records contain security event information for a system. Such information can include successful and failed authentication attempts, file accesses, security policy changes, account changes, and use of privileges. As with system events, operating systems typically permit system administrators to specify which types of events should be

audited and whether successful and/or failed attempts on various system actions can be logged.

Operating system logs may also contain information from other software and applications running on the system. System logs are useful for identifying and investigating suspicious activity on a particular system. System logs are often analyzed to compile information on suspicious activity. For example, in the 2009 CERT guide: *Common Sense Guide to Prevention and Detection of Insider Threats 3rd Edition*, the majority of insider threat participants were identified via system logging [11].

System logging on the Linux operating system originates with its derivation of the Unix operating system and its applicable function of the same name. The necessity for such a logging system became a necessity as operating systems and applications grew more complex. Systems were devised to categorize and log the multitude of messages and allow the operations staff to quickly analyze problems from simple status messages.

2.7.2. Syslog

The BSD Syslog Protocol is one such system that has been widely accepted in many operating systems [29]. Flexibility was designed into this process so administrators have the ability to configure the destination of messages sent from the processes running on the device. Events received by the syslog process can be logged to different files and also displayed on the console of the device, as well as forward the messages across a network to another syslog process on a different machine.

Basically, the syslog protocol provides a transport to allow a machine to send event messages across IP networks to syslog servers, normally over a UDP connection [29]. The protocol is only designed to transport the event messages and does not address

security or integrity of the message transmission. Using syslog, a remote host can keep track of the general well being of any other host running the syslog protocol. In an enterprise system using the syslog protocol, each message generator uses a common high-level format for its logs and the same basic mechanism for transferring its log entries to a syslog server running on another host [28]. Any operating system or application can use syslog as it provides a simple framework for message entry generation, storage, and transfer if designed properly. Many applications and operating systems either use syslog as their native logging format or offer features that allow their log formats to be converted to syslog format, such as Microsoft Windows and Cisco operating systems. An example of syslog output can be seen below in Figure 2.6.

```
-----
Feb 12 14:53:34 ubuntu cron[914]: (CRON) INFO (pidfile fd = 3)
Feb 12 14:53:34 ubuntu cron[947]: (CRON) STARTUP (fork ok)
Feb 12 14:53:34 ubuntu cron[947]: (CRON) INFO (Running @reboot jobs)
Feb 12 14:53:34 ubuntu anacron[934]: Normal exit (0 jobs run)
Feb 12 14:53:34 ubuntu NetworkManager: Ifupdown: get unmanaged devices count: 0
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): carrier is OFF
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): new Ethernet device (driver: 'vmxnet')
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): exported as /org/freedesktop/NetworkManager/Devices/0
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): now managed
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): device state change: 1 -> 2 (reason 2)
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): bringing up device.
Feb 12 14:53:34 ubuntu kernel: [ 29.458445] ADDRCONF(NETDEV_UP): eth0: link is not ready
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): preparing device.
Feb 12 14:53:34 ubuntu NetworkManager: <info> (eth0): deactivating device (reason: 2).
Feb 12 14:53:34 ubuntu NetworkManager: Added default wired connection 'Auto eth0' for
/sys/devices/pci0000:00/0000:00:11.0/0000:02:01.0/net/eth0
Feb 12 14:53:35 ubuntu NetworkManager: <info> modem-manager is now available
Feb 12 14:53:35 ubuntu NetworkManager: <WARN> default_adapter_cb(): bluez error getting default adapter: The
name org.bluez was not provided by any .service files
Feb 12 14:53:35 ubuntu NetworkManager: <info> Trying to start the supplicant...
Feb 12 14:53:35 ubuntu gdm-binary[743]: WARNING: Unable to find users: no seat-id found
Feb 12 14:53:35 ubuntu gdm-simple-slave[965]: WARNING: Unable to load file '/etc/gdm/custom.conf': No such file
or directory
```

Figure 2.6. Sample Syslog Output.

Syslog assigns a priority to each message based on two attributes, the message type or facility and its severity level. The facility attribute is composed of 24 facility (typically only 18 are used) types ranging from kernel messages to eight locally defined categories. Other facility types are mail system messages, authorization messages, printer messages, and audit messages. Severity levels are divided into eight categories

ranging from emergency, where the system is unusable, to debug-level messages [29].

The facilities and severity levels can be seen in Table 2.3 [29].

Table 2.3. Syslog Facility and Severity Levels.

Facility	Message Description
kern	Kernel messages
user	User-level messages
mail	Mail system
daemon	System daemons
auth	Authorization system: general security events
syslog	Syslogd messages
lpr	Line printer system
news	News subsystem
uccp	UUCP system.
cron	Clock daemon
Authpriv	Security/authorization messages: access-control related messages
ftp	FTP daemon
Ntp	NTP subsystem
local0-7	Locally defined categories
mark	Generated by syslog itself for timestamping logs.
Severity	Description
emerg	Emergency: System is unusable
alert	Alert: Action must be taken immediately
crit	Critical: Critical conditions
err	Error: Error conditions
warning	Warning: Warning conditions
notice	Notice: Normal but significant condition
info	Informational: Informational messages
debug	Debug: Messages for debugging purposes

Syslog can be configured to handle log entries based on each message's facility and severity. For example, it could forward emergency-level (0) kernel messages to a log server for further analysis, and record all debug-level (7) messages without forwarding them. This is the extent of the protocol's message handling capability, as it cannot make decisions based on the source or content of a message [28].

Syslog was designed to be very simple with a three-section format for messages. The first section specifies the facility and severity as numerical values, commonly called the priority value. The second section of a syslog message is a timestamp and the

hostname or IP address of the message source. The final section is the actual message content as defined by the generator of the message, since no standard fields are defined within the message content. The messages are intended to be human-readable, and therefore not easily machine-readable. This simple design provides flexibility for log generators, which allows whatever information deemed important to be placed in the content field [29].

Given that syslog messages were designed to be human-readable, the sheer number of messages generated in systems today is nearly impossible to analyze manually. To increase analysis efficiency, log parsing tools have been developed to aid administrators. Regex scripting and perl scripting are two such examples along with other log parsing programs developed by various third-party entities.

2.7.3. Rsyslog

Rsyslog is a general public licensed, enhanced version of the syslog protocol. It supports all the functionality and settings of syslog, but also offers support for reliable syslog over TCP, writing to MySQL databases and fully configurable output formats including timestamps. Rsyslog was initiated by Rainer Gerhards. In 2004, rsyslog was split from the sysklogd development team with the goal to provide a more feature-rich and secure syslog daemon while retaining drop-in replacement capabilities to stock syslogd implementations. Rsyslog is more secure by natively supporting reliable transmission modes like TCP or RFC 3195 (syslog-reliable) [30].

The rsyslog daemon supports an enhanced syslog configuration file format, and also works with the standard syslog configuration file. It is possible to simply replace the syslogd binary with the one that comes with rsyslog, although modifications to the

configuration file would be required to use the newer features of rsyslog over syslog. Rsyslog was released in February 2007 and development is still continuing. As of today, rsyslog is the default implementation of system logging in two of the more popular Linux distributions, Fedora and Ubuntu, as of versions 8 and 9.10 respectively.

2.7.4. Audit Subsystem

At the system level, auditing refers to the logging of the actions of users and programs of a system. The Linux Audit daemon performs the latter form of auditing. As a passive security measure it only detects, not enforce violations of security policy. Because the kernel provides the lowest-level access to system resources, every application must make a system call, either directly or indirectly, the audit daemon is capable of monitoring all system calls made to the kernel. Additionally, the audit daemon integrates with Security-Enhanced Linux (SE-Linux) and can log violations of SE-Linux policy [31].

Linux auditing helps secure a system by providing a means to analyze system actions in great detail. Audit subsystem consists of several components contributing to the functionality of the overall framework. The audit kernel module intercepts the system calls and records the relevant events. The auditd daemon writes the audit reports to disk. Various command line utilities take care of displaying, querying, and archiving the audit trail [32]. Figure 2.7 shows a sample audit record and Figure 2.8 [32] shows the components involved in the Linux Audit Subsystem.

The audit subsystem is comprised of seven components in a Linux distribution. The audit daemon (auditd) writes the audit messages to disk that are generated via the audit kernel interface, which are triggered by application and system activity. The audit

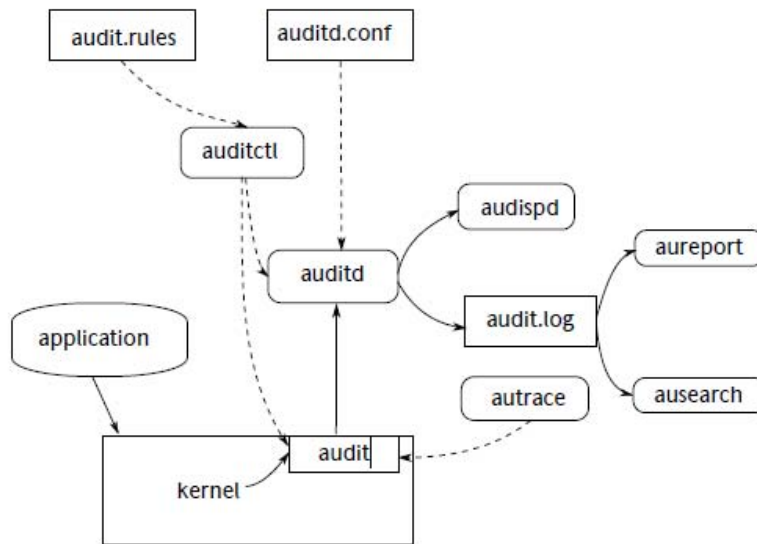
daemon is started and controlled by its configuration file, normally located at `/etc/sysconfig/auditd`. The audit functions are controlled by `/etc/auditd.conf` once the daemon is started.

The `auditctl` utility controls the audit system. Log generation parameters, kernel settings of the audit interface, and rule sets that determine which events are tracked are controlled by this utility. The file `/etc/audit.rules` contains a list of `auditctl` commands that are loaded at system boot time immediately after the audit daemon is started. These rules are what allow an administrator to custom tailor what events the subsystem may monitor.

The `aureport` utility provides the capability to create custom reports from the audit event log. This report generation can be scripted and the output used by other applications. The `ausearch` utility provides the ability to search the audit log file for certain events using characteristics of the logged format or keywords provided in the audit rules. The audit dispatcher daemon (`audispd`) allows the subsystem to relay event notifications to other applications instead of or in addition to writing them to disk in the audit log if necessary. The `autrace` utility can trace individual processes and is similar to `strace`. The output of `autrace` is logged to the audit log. [32]

```
time->Wed Nov 18 08:34:10 2009
type=CONFIG_CHANGE msg=audit(1258562050.969:36): auid=4294967295 ses=4294967295 op="add rule" key="syslogall"
list=4 res=1
----
time->Wed Nov 18 08:58:36 2009
type=PATH msg=audit(1258563516.287:78): item=0 name="syslog.all.log" inode=83 dev=08:01 mode=0100640 ouid=101
ogid=4 rdev=00:00
type=CWD msg=audit(1258563516.287:78): cwd="/var/log"
type=SYSCALL msg=audit(1258563516.287:78): arch=400000003 syscall=85 success=no exit=-22 a0=bf8c36ec a1=bf8c46ec
a2=fff a3=1 items=1 ppid=2896 pid=2907 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts0 ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="syslogall"
```

Figure 2.7. Sample Audit Log Entry.



Straight arrows represent the data flow between components while dashed arrows represent lines of control between components.

Figure 2.8. Linux Audit System Components.

The audit subsystem traces processes to the user ID that started them. This allows an administrator to trace which user owns which process and determine who is potentially performing malicious operations. The Linux audit subsystem also provides tools that write the audit reports to disk and translate them into human readable format. Linux auditing provides the capability to filter the audit reports for certain events of interest, by filtering on the following data:

- User
- Group
- Audit ID
- Remote Hostname
- Remote Host Address
- System Call
- System Call Arguments
- File

- File Operations
- Success or Failure

Linux auditing provides the means to filter the audit reports for events of interest and to record selected events. It allows the creation of customized rule sets to direct the audit daemon to record events of interest. In the subsystem, audit reports are owned by root and unauthorized users cannot remove the audit logs. If an audit event cannot be recorded, the audit subsystem can trigger a shutdown of the system to keep events from escaping its control. Since this shutdown would be an immediate halt of the system without any syncing of the latest logs to disk, the default configuration for this capability is to log a warning to syslog rather than to halt the system. The subsystem allows a clean shutdown, if the system runs out of disk space when logging, but the default configuration is to stop logging when there is no more disk space. [32]

2.8. Summary

This chapter has covered the necessary background for this research study. The objectives for maintaining information security and insider threat trends, characteristics, and types have been covered. The basics of what auditing is and its current uses and risk management basics were also described. Finally the types of logging and auditing in a Linux operating system were described at an overview level.

III. Methodology

This chapter outlines the methodology used to develop use cases based on a risk assessment methodology and create a customized auditing policy for a Linux workstation. The problem definition and research goals are presented and followed by taxonomy of insider threat and an associated risk assessment process. The chapter finishes with how an organization would customize the system logging and auditing rule sets for a Linux workstation.

3.1. Problem Definition

While most system security tools deal with outside attacks, organizations have recognized the importance of insider threat. To help detect and mitigate such attacks organizations may implement specialized tools, take advantage of available resources, or ignore such risks. Organizations may use the existing auditing capabilities to provide a layer of defense to the insider threat without ignoring the risks or purchasing costly proprietary solutions. Although the Defense Information Systems Agency (DISA) does offer a Security Technical Implementation Guide (STIG) for Unix operating systems that delineates some auditing rules, its main purpose is for general system security with no specific objective to monitor possible insider threat [33]. This does not address an organization's unique security risks and vulnerabilities, especially if such an organization utilizes a Linux distribution other than RHEL5. Rather than rely on a general workstation auditing configuration, an organization should know exactly what events its workstations need to audit based on the organization's computer security requirements

and priorities. Currently there exists no defined methodology for creating a customized Linux auditing template based on organizational security requirements and priorities.

3.1.1. Research Goals

The main goal of this research is to establish a methodology for developing an organization-specific auditing configuration for a Linux workstation to detect insider threat. This research should delineate the need for effective risk assessment based on an organization's needs and priorities to determine use cases that will help create a tailored auditing configuration based on an organization's security requirements and risk assessment. This research uses the existing system logging and auditing capabilities of a Linux workstation and tailors it to detect activities associated with malicious insider behavior.

3.1.2. Expected Outcome

It is expected that once the proof-of-concept method is developed, organizations will be able to more effectively and efficiently detect possible insider threat activity. In turn, this should allow organizations to allocate scarce security resources to more effectively mitigate insider threat risks.

3.2. Approach

An evaluation of the auditing and logging capabilities available for configuration in Linux is performed to determine whether insider threat scenarios developed by a risk assessment methodology can be detected. A series of 24 malicious insider scenarios or use cases, developed from risks determined by King's extended ITFDM, are simulated and the logs resulting from these activities are analyzed. The end result is a customized

auditing and logging configuration based on an organizations perceived threats and vulnerabilities.

3.3. Extended ITFDM

King's vulnerability assessment methodology starts by performing a content analysis of existing insider threat research, case studies, and surveys. This analysis provided a way to determine the more prevalent actions performed by insider threats. The methodology includes analysis of a variety of sources to determine insider threat commonalties. The analysis of a number of recent sources to create a more comprehensive list of various actions and is a key component of this methodology.

Insider threat actions identified in the content analysis phase were categorized according to possible insider threat intentions and resulted in a more comprehensive intent-based ITFDM (Insider Thread Functional Decomposition Model). This extended model listed each applicable action under the various intentions.

Before determining insider threat intentions, the insider threat actions identified in the content analysis are categorized into the four states in the ITFDM (Figure 3.1) [3]. Each insider threat action is analyzed and re-classified to determine the insider's primary intent. After identifying the intention of each action, those sharing similar intentions are grouped together.

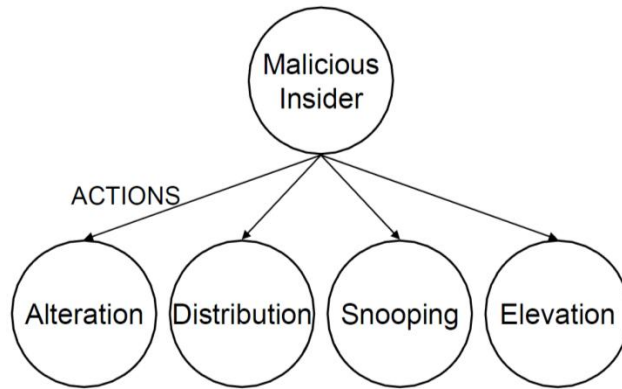


Figure 3.1. Root and 1st Tier ITFDM Action Leaf Nodes.

Figure 3.2 [3] shows the sixteen insider threat intentions identified in King’s research. Figures D.1-D.4, map each action to the insider threat’s applicable intention from King’s research (located in Appendix D [3]).

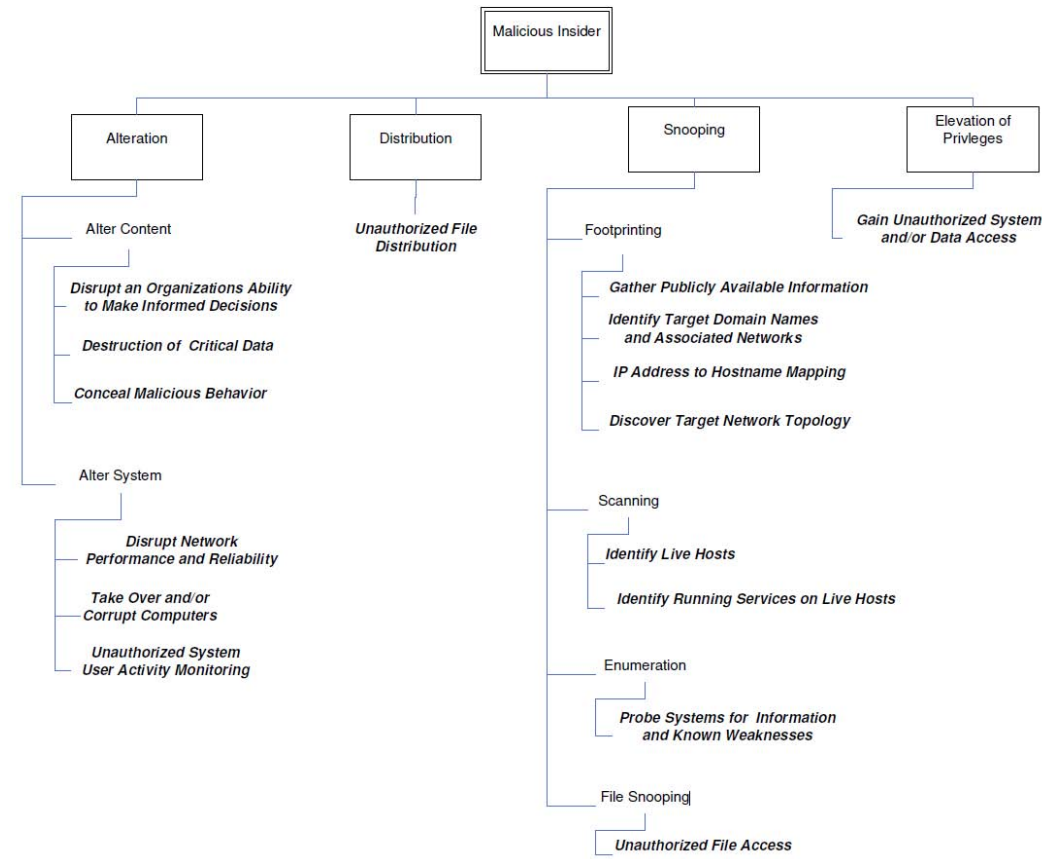


Figure 3.2. ITFDM Malicious Intentions.

King proposes a periodic update of the intent-based ITFDM annually to preserve the utility of the extended model to ensure it contains current vulnerabilities. Classifying insider actions based upon primary intention allows organizations to focus on their primary areas of concern and associated actions. Depending on an organization's requirements and priorities, some intentions and associated actions are more applicable than others. King's extended ITFDM allows organizations to focus on specific insider threat intentions of concern and actions commonly used to exploit those intentions.

3.4. Insider Threat Use Case Detection Verification

The System Under Test (SUT) in this study is the Linux auditing subsystem and logging subsystem running inside a virtual computer, using VMWare Workstation. It is a basic installation of Ubuntu version 9.10 with all security patches as of 23 November 2009. The SUT includes adjustable auditing and logging configurations that collectively make up the Component Under Study (CUS).

The input for the SUT is the group of 24 insider threat use cases. All scenarios are manually executed through the SUT, one at a time while the CUS audits or logs for success and failure. The Linux subsystems tested recording of detectable events in the audit log or appropriate syslog file for each subsystem respectively. After each test run is complete, the virtual machine is reset to the baseline configuration. This is accomplished using a VMWare snapshot.

The scope of the study is limited to a single workstation operating in VMWare without network connectivity. It is designed to simulate the security implications of multiple users sharing a common workstation. The workstation does not include the

logging of network access and interaction with other workstations, servers, or peripheral devices.

3.4.1. System Logging Configuration

The parameters for the logging subsystem are the 24 facility groups and eight severity levels configured to output to individual files for each combination of a single facility and a single severity level. For example log messages with the kernel (0) facility and the emergency (0) severity level would be outputted to the log file at `/var/log/rsyslog.kernel.emerg` on the Linux workstation. This will result in 192 individual log files. The rsyslog configuration will not change for each use case scenario. All system log files are cleared when the SUT is reset by VMWare SnapShot before each use case is tested.

3.4.2. Auditing Configuration

The parameters for the auditing subsystem are the system call and file watch settings configured to audit success and failure of the pertinent system call and/or file described in each use case. The auditing subsystem can track any kind of file system access to important files, configurations or resources. It allows the addition of watches on these and can assign keys to each kind of watch for better identification in the logs. The system call auditing functionality allows the tracking of the system's behavior below the application level with the option of filtering auditable events with a great deal of granularity. The output of each use case is moved from the `/var/log/audit/audit.log` default location to a separate file for each use case. The audit log is then cleared when the SUT is reset by VMWare SnapShot before each use case is tested.

3.5. Data Collection

The output of the SUT for each scenario is annotated in a Microsoft Excel® worksheet as a success or failure of detection and grouped by the log type (rsyslog, audit, or both). The pertinent log files are saved in .txt format on the host system. The associated system or audit message events key to detecting each use case are also annotated on the Microsoft Excel® worksheet for possible future research.

3.6 System Setup

The virtual machine consists of VMWare Workstation, version 6.5.3, build 185404. The hardware platform is an AOpen mini-PC, Model MP945-D, with a 2.20 GHz T2450 Intel processor, and 2 GB of RAM. The Virtual Operating System is a standard Ubuntu distribution, version 9.10, with all available security patches downloaded and installed as of 23 November 2009.

3.7 Procedure

Each time a use case scenario is simulated, the following steps are completed to ensure that the obtained samples are independent and specific to the current scenario:

1. Load VMWare SnapShot – This resets the virtual machine to the baseline for each scenario ran.
2. Configure Audit Settings – The rsyslog.conf configuration file is constant for every use case and does not change as the configuration allows for the capture of all system log messages for every facility at every severity level. The audit.rules file is configured with the appropriate file watches and/or system call watches as determined by the use case. The audit subsystem is then restarted with newly configured rule set.

3. Wipe Log Files – All logs (all system logs and the audit log) are cleared to ensure that there are not any other logged events in them.
4. Perform Use Case – The 24 use case scenarios are manually executed using a checklist of exactly what is to be performed.
5. Save Log Files – Upon completion of the use case, all logs (system logs and audit log) are saved as .txt files on the host system for later analysis offline.
6. Evaluate Log Files – Upon completion of the use case scenario, the log files are evaluated to determine if any identifying events for the use case have been logged. These key events are separated by scenario and annotated on the aforementioned Microsoft Excel® worksheet for further analysis.

3.8. Use Case Scenarios

A critical factor in the methodology is the selection of use case scenarios. An organization must select carefully to ensure a comprehensive representation of their insider threat security requirements. For this study, the scenarios are selected based upon possible requirements of a typical organization using the King's extended ITFD model. The 24 use cases are general examples of possible scenarios an organization might select. The scenarios focus on protecting the confidentiality of files on a shared workstation. The scenarios simulate multiple users using a common workstation, and a malicious user attempting to access other user's files. The scenarios simulate various techniques that a malicious insider might use to obtain access to vital information. Privileges held by administrators as well as normal users are represented in the scenarios. The derivation of these use cases is presented in the next chapter (Table 4.1).

3.9. Evaluation Technique

Once the results from the use case scenarios are collected and evaluated, they are put into a table for ease of analysis to determine what system log settings and audit rule sets can identify insider threat activities for an organization based on their security requirements. This table will enable an organization to determine what additional control measures are necessary to identify insider threat risks identified that were not detected by the Linux logging subsystem nor the audit subsystem.

3.10. Summary

The methodology and verification described in this chapter can be used to create a customized insider threat auditing template for a Linux workstation. It is used to determine how well the Linux logging and auditing subsystems can detect particular insider threats, using scenarios focused on unique organizational security requirements. The results of a notional example are presented in the next chapter.

IV. Use Case Verification

This chapter presents a notional example to demonstrate the application of the concepts presented in this research to include: (1) the intent-based ITFDM, (2) verification of insider threat detection via auditing, and (3) development of an organization-specific auditing and logging rule set for insider threat detection.

4.1. Notional Example

To demonstrate the concepts proposed in this research and understand their potential, a notional example is developed. A generic organization is defined to demonstrate the wide applicability of the methodologies without limiting them to any one type of organization (e.g., commercial sector or Department of Defense).

4.1.1. Example Overview

For the purpose of this example the organizational leadership wishes to determine what sort of security policy is needed to deter or detect insider threat. The organization is concerned with the security of its stand-alone systems, which have no outside connectivity. These systems are located in secure facilities with administration personnel located off-premises. Due to the fact that administration personnel are not co-located with the systems in question, personnel have terminal window access to perform basic system administration functions at the direction of IT support personnel.

The primary concern of the organization in regards to insider threat is the average employee with basic knowledge of the Linux system. Such concerns are readily supported by recent insider threat studies conducted by CERT [11]. The most recent study found that a majority of insider threat actions were concerned with theft for

business or financial gain. In 85% of theft for financial gain cases studied, the insiders used their own username and password and in all the cases involving theft for business gain none of the insiders had neither system nor database administration access [11]. While IT sabotage is an organizational concern, the sabotage risks involved with stand-alone systems is deemed to be acceptable by the organization.

4.1.2. Example Problem Definition

Suppose an organization must secure stand-alone workstations and leadership has requested a security policy pertaining to insider threat to complement existing security policy. The organization's objective is to assess the effectiveness of existing system capabilities and to obtain an analysis as to the effectiveness of such measures.

4.2. Methodology Demonstration

This section explains how the methodology described in chapter III would be conducted to provide a customized auditing policy to detect insider threat. This methodology will start with creating use cases by utilizing the ITFDM discussed as a risk assessment methodology. Other risk assessment methodologies may be used, but for the purpose of this study, King's intent-based ITFDM is utilized [3]. Once the use case scenarios have been created, the use cases are verified on a test system. Once all the use cases have been tested, an analysis of such results is conducted to determine what scenarios can be detected and what key event messages are generated to denote suspicious behavior.

4.2.1. Extended ITFDM use case scenarios

The use case scenarios are developed using the extended ITFDM proposed by King. In this method a content analysis of current research in prevalent insider threat

cases was performed. Such research includes the CSI/FBI annual studies and the insider threat studies performed by CERT and the US Secret Service [11][14-17][19][34-41]. Given the scope of the problem definition and organization priorities, 24 use case scenarios were developed for test verification.

The categorization proposed by King allows for a more granular examination of insider threat risk by intention, allowing an organization to properly assess impact of risks determined by the content analysis, because they were developed with the organization's specific limitations in mind. For example with the system specified as a stand-alone, file distribution threats would be limited to mounting external media to the workstation, such as a USB flash drive, and may be deemed to be a lower acceptable risk than an insider's unauthorized access to critical data. Keeping in mind such factors allows an organization to properly define and assess the risk and impact of the possible insider attack vectors presented from the content analysis.

4.2.2. Use Case Scenarios

These use cases were developed by a reasoned discussion in an insider threat working group composed of security and IT support personnel. This group theorized what actions an average user may take depending upon the intent of the malicious insider in addition to the content analysis of recent insider threat trends as dictated by King's methodology. It is assumed the group will utilize previous use cases documented by attack trees to refine possible attack paths over time and benefit the development of new use cases with periodic re-evaluations of the system. A list of the 24 use cases developed for this example can be seen in Table 4.1 and detailed analysis of these cases are located in Appendix E.

4.2.2.1. Gain Unauthorized System and/or Data Access

Pertaining to an insider's intent to "gain unauthorized system and/or data access," an insider attempting to gain root access was determined to be unlikely but very damaging. Although a low probability was assessed to this risk, the possible impact to needed to be addressed. Given a hypothetical situation in which a malicious insider may attempt to gain root access and perform unauthorized actions on critical data. It was determined the most probable methods would be through the use of the *su* and/or *sudo* commands. Current policy prohibits the use of the *su* command and requires the use of the *sudo* command for any root level actions performed by system administrators, therefore it was determined that any observable events denoting the use of the *su* command could be considered suspicious, while any event denoting the use of the *sudo* command would require investigation.

While root-level access affords an insider with absolute power over a Linux workstation, it was determined that monitoring all root-level actions would be overwhelming to security and administrator personnel. The limitations of the system logging and auditing functions of the Linux operating system would require monitoring all actions performed at the root level, since differentiating between user and automated kernel actions is non-existent in the operating system.

4.2.2.2. Conceal Malicious Behavior

In the unlikely event an insider was to gain root level access, the organization wanted to determine if malicious actions pertaining to system logging and auditing could be monitored. Given the default restrictions on the logging and auditing systems' configurations as well as the logs, it was determined that user attempts would normally

be unsuccessful, but any attempt to conceal malicious behavior should be tested for observability.

If an insider were to operate within the organization, hiding his or her actions would be paramount to escape detection either as root or a normal user. Therefore eight use cases were developed to determine if such actions could be observed with the logging and auditing subsystems. The use cases developed to test for the reading, modification, or deletion of the configuration files and logs at the user, administrator, and root privilege levels. A use case was also created to test if a user could be observed attempting to disable the logging and auditing daemons. In addition, a seemingly benign task of clearing a user's own *.bash_history* file could be considered an attempt to conceal malicious behavior. While such a command history file would be of little use in event reconstruction with its limited storage, the fact that a user would hide such behavior might be a behavioral indicator for insider activity.

4.2.2.3. Take Over and/or Corrupt Computers

It was also determined that an insider would attempt to escalate account privileges to access critical information by attempting to modify system files such as the */etc/group* or */etc/sudoers* file. Attempts to add additional system accounts by unauthorized personnel should be monitored as well.

4.2.2.4. Unauthorized File Access

The organization leadership in this example determined that two files and the directories that contain them as critical information and should be monitored for unauthorized access. Monitoring of critical data was to determine if unauthorized attempts to access the information could be observed. Seven use case scenarios fall within

this category to reflect the large percentage of insider cases relating to theft discussed earlier in this chapter (Section 4.1.1. Example Overview). Given that administrator level access did not forgo standard owner and group privileges unless root level access was invoked, the majority of these use cases pertain to the attempts made at the standard user level. The use case results would help determine what observables are generated when a user attempts to access a critical file with and without group access as well as attempts to change directory or file attributes.

4.2.2.5. Unauthorized File Distribution & Destruction of Critical Data

The last two use cases pertain to file distribution situations and destruction of critical data. The first use case of the two determines if an outside storage device is mounted to the file system. As the system has no outside connectivity, an external storage device is the only way for a file to be distributed. The second use case determines if an unauthorized attempt to delete critical data can be detected.

4.2.3. Verification Analysis and Results

It was determined during the verification of the results that the Linux system logging system is not capable of monitoring file access events. The only situations deemed useful for the purpose of this research was to monitor the use of the *su* and *sudo* commands. The event messages generated allowed for security personnel to determine who attempted to initiate the commands and their success or failure in the attempt. In the case of the *su* command, the auditing system could monitor successful attempts. But unsuccessful attempts in this situation are equally useful in targeting suspect individuals.

Table 4.1. Insider Threat Use Cases.

1	User with root password attempts to use 'su' command (Security policy disallows root login and the use of 'su' command)
2	User without root password attempts to use 'su' command (Security policy disallows root login and the use of 'su' command)
3	Authorized user attempts to use 'sudo' command without valid password (up to 3 unsuccessful attempts)
4	Unauthorized user attempts to use 'sudo' command with valid password
5	Admin/root attempts to access User directory and file
6	Admin/root attempts to modify system logging and audit rule sets
7	Admin/root attempts to modify system logs and audit log
8	Admin/root attempts to clear system logs and audit log
9	User attempts to read system logs and audit log
10	User attempts to modify system logs and audit log
11	User attempts to clear system logs and audit log
12	User attempts to stop/disable rsyslog and auditing daemon
13	User attempts to modify system log and audit rule sets
14	User attempts to add system account
15	User attempts to modify user/group information <i>/etc/group,passwd,gshadow,shadow,/security/opasswd</i>
16	User A attempts to access User B directory
17	User A attempts to access User B file w/o group access
18	User A attempts to access User B file w/ group access
19	User A attempts to modify permissions of User B directory(chgrp, chown, chmod)
20	User A attempts to change permissions of User B file w/o group access (chgrp, chown, chmod)
21	User A attempts to change permissions of User B file w/ group access (chgrp, chown, chmod)
22	User attempts to mount an external USB flash drive
23	User attempts to clear <i>.bash-history</i> file
24	Unauthorized user attempts to delete a file

In the unlikely event that an insider gained root privileges, the auditing subsystem could detect accesses to critical data and monitor the access or modification of logging and auditing configurations or logs. In the instance of deleting the system logging or auditing logs, the auditing subsystem could detect the deletion of the system log, but not its own audit log. After analysis of this use case, it was determined that deleting either

log would generate a gap in monitoring coverage that could be noted by review of the existing logs by security personnel and therefore monitoring by the system itself is not required

All attempts where a normal user attempts to modify or disrupt the logging or auditing capabilities of the workstation proved to be unsuccessful. All unsuccessful attempts were observed with the exceptions of reading/modifying the system and audit logs themselves and modifying the audit rule set. Such attempts were denied by the system due to permissions, but the attempts did not generate any event messages in either subsystem. A normal user could view the rsyslog configuration, but this would not provide any insight into whether insider threat was being monitored.

User attempts to modify system files such as */etc/group* and */etc/sudoers* are unsuccessful due to permissions. Of these use case scenarios, generated events only occurred when a user had read permission to the system file, but no write access and a forced write was attempted on the file. For example, all users have access to read the */etc/group* file to determine group membership of system accounts. When this file is opened within an editor a successful event message is generated if read access is monitored, and an unsuccessful event message is generated when a write attempt occurs. No observable event messages were generated when a user attempted to add a system account using the command line, as the command is only executable as an administrator or root. This is because the command does not perform a write command to any system file before it authenticates the user account attempting to add a user to the system.

Finally, successful and unsuccessful attempts to access critical data directories and files were generally detectable with one caveat. Event messages were not detectable

in which an insider attempted to perform an action on a file within a restricted directory. As the audit rules pertain to the specified file, the directory permissions prevented the insider from “touching” the restricted file. An in-depth analysis and description of each use case is presented in Appendix E.

4.3. Organization Specific Auditing Policy

After analyzing the results of the use case scenario verification testing, the organization may now formalize a specific auditing policy pertaining to its stand-alone Linux workstations. This policy includes the system log configuration and auditing rule set along with the key event messages to detect such actions. In this notional example such a policy would include the possible insider actions listed in Table 4.1.

4.3. Auditing and Logging Limitations

While the auditing and logging systems could detect most of the use cases, care must be taken to remain aware of the limitations of the systems in question. The system logging system has limited capability in detecting insider threat. Its primary use is to detect attempts to gain unauthorized system access by using the *su* or *sudo* commands. Although the use of the *su* command could also be detected by setting a file watch for execution of the */bin/su* file, or use a file watch to monitor the */var/run/utmp* database for writes to determine if root sessions have been opened. The later file watch would only detect successful attempts and may not be conducive to detecting possible threats.

Table 4.1. Notional Example Logging and Auditing Policy for Insider Threat.

Logging/Auditing Requirement	Configuration Setting
Monitoring via system logging of <i>su</i> and <i>sudo</i> command executions [Configuration setting for rsyslog.conf]	Authpriv.=info /var/log/insider.log Authpriv.=err /var/log/insider.log Authpriv.=notice /var/log/insider.log Authpriv.=alert /var/log/insider.log
Audit root actions in accessing critical information files	-w /home/user1 -p rwx -w /home/user1/file1 -p rwx -w /home/user3 -p rwx -w /home/user3/file3 -p rwx -w [critical data dir path] -p rwx -w [critical data file path] -p rwx
Audit root actions in accessing and modifying system logging and auditing functions	-w /etc/audit -p rwx -w /etc/audit/auditd.conf -p rwx -w /etc/audit/audit.rules -p rwx -w /etc/rsyslog.conf -p wx -w /var/log -p rx -w /var/log/insider.log -p rx -w /var/log/audit -p rx -w /var/log/audit/audit.log -p rx
Audit user attempts to remove/delete system logs	Handled by audit rules for root actions
Audit user attempts to disable system logging and auditing	-a exit,always -S kill -F path=[audit daemon path] -F uid>=1000 -a exit,always -S kill -F path=[rsyslog daemon path] -F uid>=1000
Audit user attempts to modify system logging configuration	Handled by audit rules for root actions
Audit user write attempts to system files	-w /etc/group -p wx -w /etc/passwd -p wx -w /etc/gshadow -p wx -w /etc/shadow -p wx -w /etc/security/opasswd -p wx -w /etc/sudoers -p wx -w [system file path] -p [rwx]
Audit user attempts to access critical information directories and files	Handled by audit rules for root actions
Audit user attempts to modify permissions to critical data directories and files	Handled by audit rules for root actions
Audit user attempts to mount external media	-a entry,always -S mount -F uid>=1000
Audit user attempts to clear account /home/[username]/.bash_history files	-w /home/[username]/.bash_history -p w
Audit user attempts to remove/delete critical data directories and files	-a exit,always -S symlink -S link -S unlink -S unlinkat -S rename -S renameat -F path=/home/user1 -a exit,always -S symlink -S link -S unlink -S unlinkat -S rename -S renameat -F path=/home/user1/file1 -a exit,always -S symlink -S link -S unlink -S unlinkat -S rename -S renameat -F path=/home/user3 -a exit,always -S symlink -S link -S unlink -S unlinkat -S rename -S renameat -F path=/home/user3/file3

Another limitation of the auditing rule sets is the configuration rules are sequential and the system checks stop after the first successful condition is met. Therefore the order in which the rules are configured can be an issue. Security personnel

must also be aware the auditing rules register events for all successful and unsuccessful accesses to a monitored file. This limitation emphasizes the fact that signatures for use cases can play an important part in automating suspicious behavior detection as the sheer number of legitimate file accesses can hinder the detection of suspicious behavior.

The auditing system allows for the monitoring of specific files, but such monitoring must be explicitly configured in the rule set as the subsystem cannot monitor newly created files without a corresponding rule. Unlike file auditing, system call auditing can be filtered by specific fields, but careful consideration of the underlying system calls of various system commands must be examined. Given that the same result can be achieved by various methods a thorough understanding of Linux system calls is needed to effectively use this capability.

4.4. Summary

This chapter provides a notional example to demonstrate the fundamental concepts presented in this research. A generic organization's insider threat vulnerabilities and risks are assessed and organized to develop insider threat use cases using a defined risk assessment methodology. These cases are then verified against a test system to determine whether such behaviors are detectable using the Linux operating system logging and auditing capabilities. The verification results are then analyzed to develop specific auditing and logging rules tailored to meet the organization's needs and priorities and determined what key events detected indicated suspicious insider behavior. The analysis also allows an organization to assess the limitations of the Linux logging and auditing capabilities.

V. Conclusions and Recommendations

This chapter summarizes the research efforts and the research goals. First, the conclusion of this research and its significance are discussed. Suggestions for possible future research efforts are then provided.

5.1. Problem Summary and Research Effort

Determining insider actions that users can perform pose a difficult problem for organizations. An organization's ability to assess insider threats is essential to an effective insider threat security policy. Mitigating and discovering insider actions is crucial in protecting an organization's information system.- A risk assessment methodology allows organizations to identify insider actions of concern and vulnerabilities specific to that organization. Organizations also need to understand the innate capabilities of its systems. Additionally, organizations need a method to assess the effectiveness of auditing and logging strategies. Knowledge of potential insider threat, as well as the effectiveness of insider threat auditing strategies will assist organizations in providing an acceptable level of information risk.

The purpose of this research was to help define a methodology to assist organizations in defending against insider threat on Linux operating systems. By analyzing current insider threat attacks an organization may bring to force the Linux operating system's auditing and logging capabilities. Such capabilities tailored to an organization's assessed vulnerabilities and priorities can create an effective auditing policy to detect and deter insider threat.

5.2. Conclusions of Research

The extension of ITFDM into an intent-based model provides organizations a framework to identify insider threat intentions and associated actions [3]. The notional example illustrated how the extended model can provide a practical means of identifying potential insider actions. In identifying these potential vulnerabilities, 24 use cases were developed using this methodology. Utilizing these use cases allows an organization to test and refine auditing and logging policies and configurations to meet its specific needs and responsibilities. Signatures were also derived from the analysis of these use case results to allow for possible automated scanning of suspicious behavior.

It can be inferred that an organization benefits from knowing what is being audited within its workstations and to ensure that the auditing complies with its security requirements. Because security requirements are fluid and dynamic, it would be prudent that once the organization has implemented this methodology into its security policy, that periodic reevaluations of its auditing template are performed to ensure that it still fits their security needs. It would behoove an organization to make it as robust as possible by creating numerous use case scenarios. The more robust the organization makes this methodology, the more effective its insider threat security policy.

Despite the effectiveness of this technical capability, the resulting policy and system configuration is not a cure-all for deterring or predicting insider threat, nor is the detection of such acts a cure-all for the growing threat from insiders. While the results of this research provide a means to detect insider threat and can provide technical characteristics of these attacks on a Linux system, the actual output of the auditing system only provides an organization with observable behavior.

It is the responsibility of the organization's security personnel to determine whether the pattern of recorded actions on the system and other behavioral indicators is in fact suspicious. As discussed in the overview of this research study the very heart of the problem of insider threat is trust. These possible attacks occur from trusted individuals who have legitimate access and authority on the system. There is yet to be developed an effective and viable automated means to determine if human actions on a system are actual attempts to attack a system from the inside. While the results of this research provide a collected and reasoned data point in determining the validity of such actions, it requires human intervention to determine if an insider attack has been attempted.

5.3. Significance of Research

Application of the methodology developed in this research provides an organization with a method to optimize its auditing and logging capabilities. This increases the organizations capability to detect and identify insider actions early enough to mitigate potential damages. It can also reduce the workload required to review system and audit logs by reducing their size in regards to insider threat. By delineating key characteristics in the logs to possible insider actions an organization increases its ability to detect malicious activity when reviewing audit logs by tailoring the recorded information to a specific need and purpose for the organization. Finally, it allows an organization to be aware of what is being audited within its information system and the effectiveness of the ability to detect insider threat.

5.4. Future Research

There are multiple areas in which this research can be expanded. As the methodology was demonstrated with simple use case scenarios involving a stand-alone

system, determining whether multiple events can be correlated into synthetic meta-events is a viable research path. Such analysis to determine whether more sophisticated insider attacks are occurring in stages or from multiple systems is a topic not addressed in this research. Another area for expansion would be to utilize the signatures developed from the presented use cases in an automated log parser to augment near real-time detection of insider threat activity on a distributed logging architecture.

Another area of possible future research is to apply the methodology to networked workstations and servers to identify and customize auditing of network related communication within an organization. For example, lessons learned in the network monitoring domains can be applied to detecting insider threats across organizational web servers using standardized tools and a common event expression framework [44]. While expanding this methodology to other components involved with an enterprise-level network, such as routers and switches is also another possible avenue of research. By applying the methodology and possibly correlating such information from other components, an improved awareness of the auditing and logging situations in those areas would allow for more comprehensive situational awareness and detection techniques pertaining to insider threat.

Combining the Linux auditing and logging capability to detect insider threat into an IDS-based application is another suggestion. By applying the methodology to outside and inside threats an organization can apply auditing and logging configurations to provide a more reliable and well-rounded security posture while automating detection and mitigation procedures into a proven automated network component. Currently research is being conducted to configure the auditing system as a sensor that can detect

certain events and send notifications to a hybrid-IDS [42]. While current research into combining the Linux audit system with IDSs have concentrated on outside attacks, tailoring these efforts to include insider threat actions is a viable research effort. Figure 5.1 [42] shows how the Linux auditing subsystem could tie into an intrusion detection system.

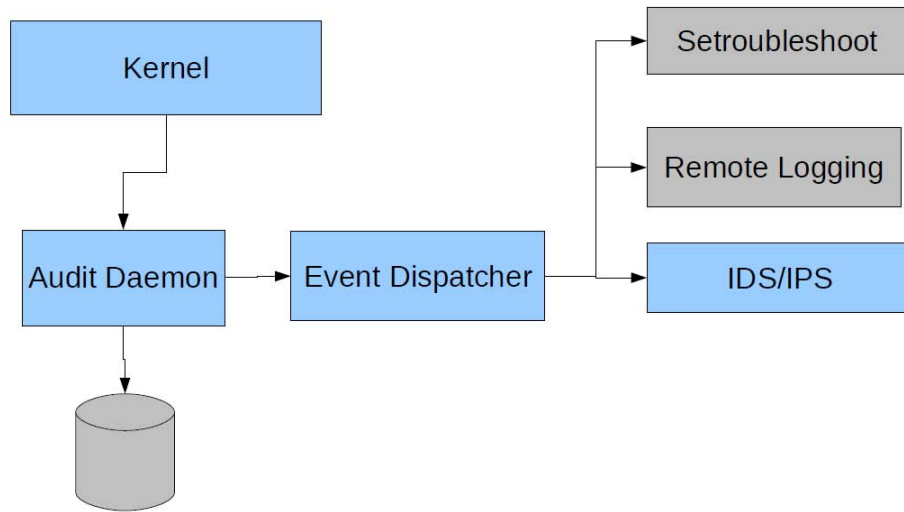


Figure 5.1. Audit System Data Flow.

5.5 Summary

This research provides organizations a methodology for customizing their auditing and logging capabilities to detect and deter insider threat. It provides a framework to create auditing rules and configure system logging to meet the needs of the organizations identified security requirements and priorities.

Appendix A: List of Acronyms

ARPANET – Advanced Research Projects Agency Network.

BSD – Berkeley Software Distribution.

CERT /CC– Computer Emergency Response Team/Coordination Center.

CIA (1) – Confidentiality, Integrity, Availability.

CIA (2) – Central Intelligence Agency.

CMU – Carnegie Mellon University.

CSI – Computer Security Institute.

CSO – Computer Security Online Magazine

CUS – Component Under Study.

USD – United States Dollars (\$).

DISA – Defense Information Systems Agency.

DHS – Department of Homeland Security.

DOD – Department of Defense.

DOE – Department of Energy.

DSS – Decision Support System.

FBI – Federal Bureau of Investigation.

FIPS PUB – Federal Information Processing Standards Publication.

STIG – Security Technical Implementation Guide.

IATAC - Information Assurance Technology Analysis Center.

IDS – Intrusion Detection System.

IP – Internet Protocol.

IT – Information Technology.

ITFDM – Insider Threat Functional Decomposition Model.

NIAC – National Infrastructure Advisory Council.

NIST – National Institute of Standards and Technology.

NTAC – National Threat Assessment Center.

RFC – Request For Comment.

RHEL5 – Red Hat Enterprise Linux version 5.

SE-Linux – Security Enhanced – Linux.

SEI – Software Engineering Institute.

SUT – System Under Test.

TCP – Transmission Control Protocol.

USSR – United Soviet Socialist Republic.

USSS – United States Secret Service.

Appendix B: NIST 800-30 Risk Assessment Methodology Flowchart

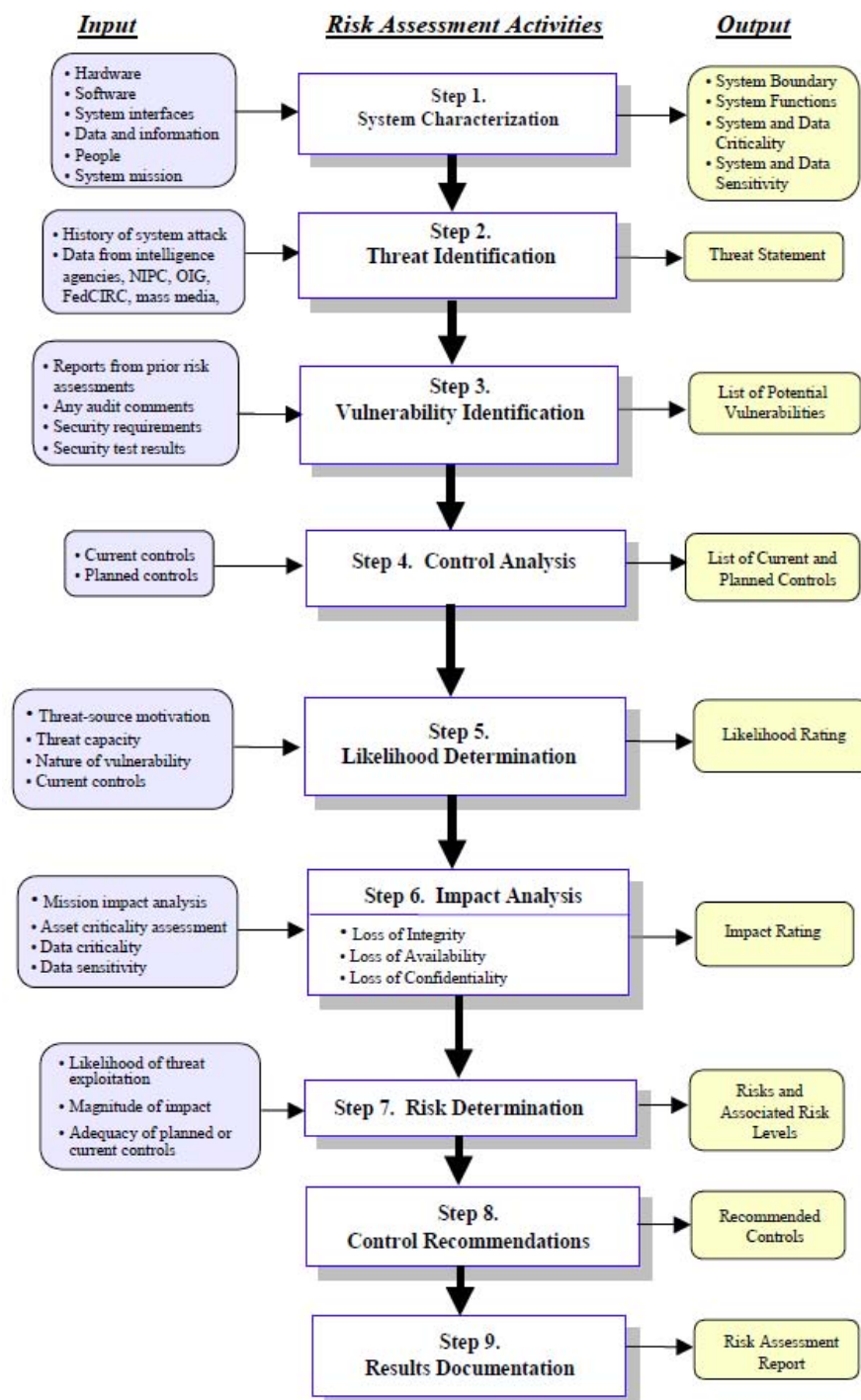


Figure B.1. Risk Assessment Methodology Flowchart.

Appendix C: NIST 800-30 Risk Mitigation Methodology Flowchart

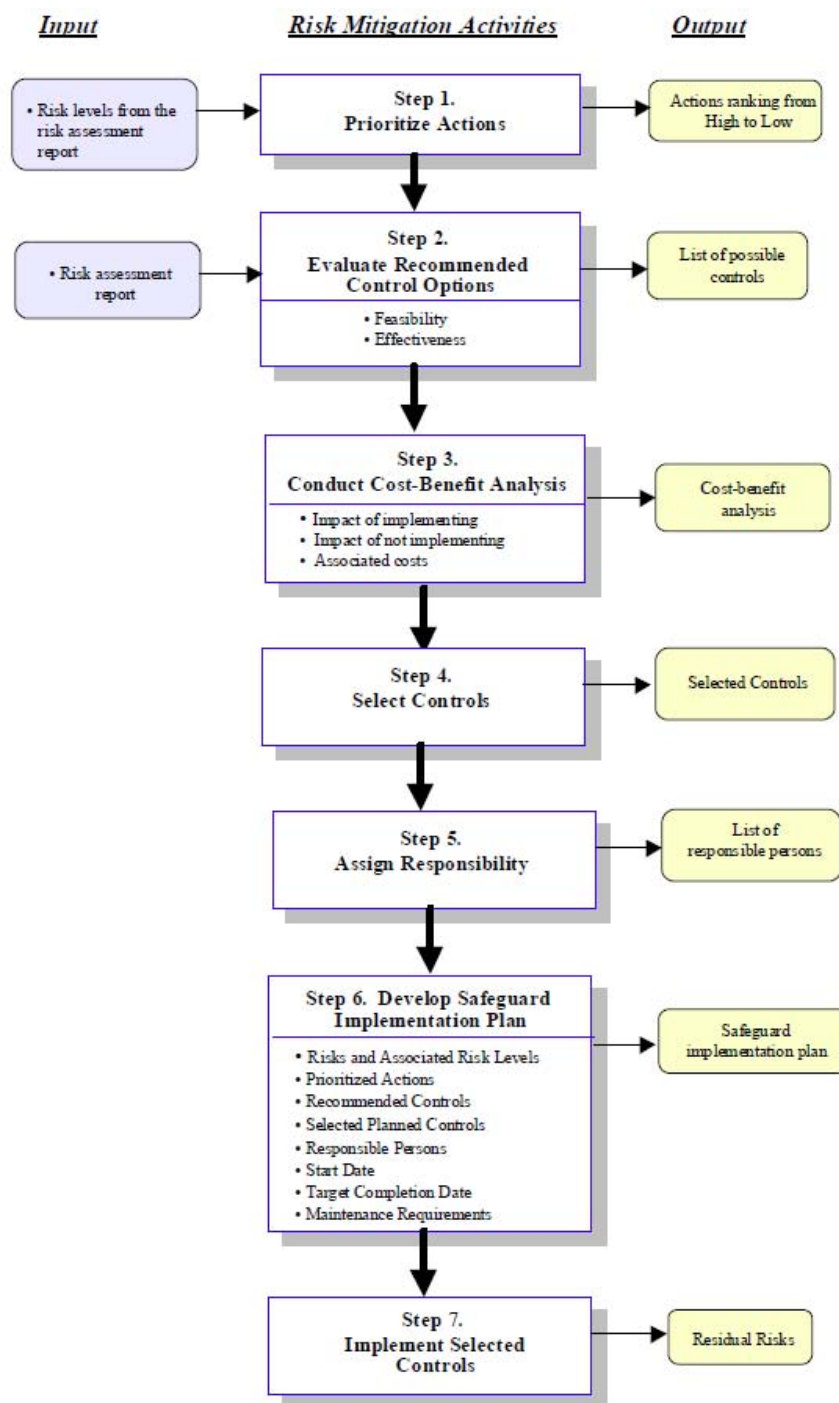


Figure C.1. Risk Mitigation Methodology Flowchart.

Appendix D: ITFDM: Action, Alteration, Snooping, and Elevation

Figures D.1 – D.4 [3] are the functionally decomposed malicious insider four protection state models.

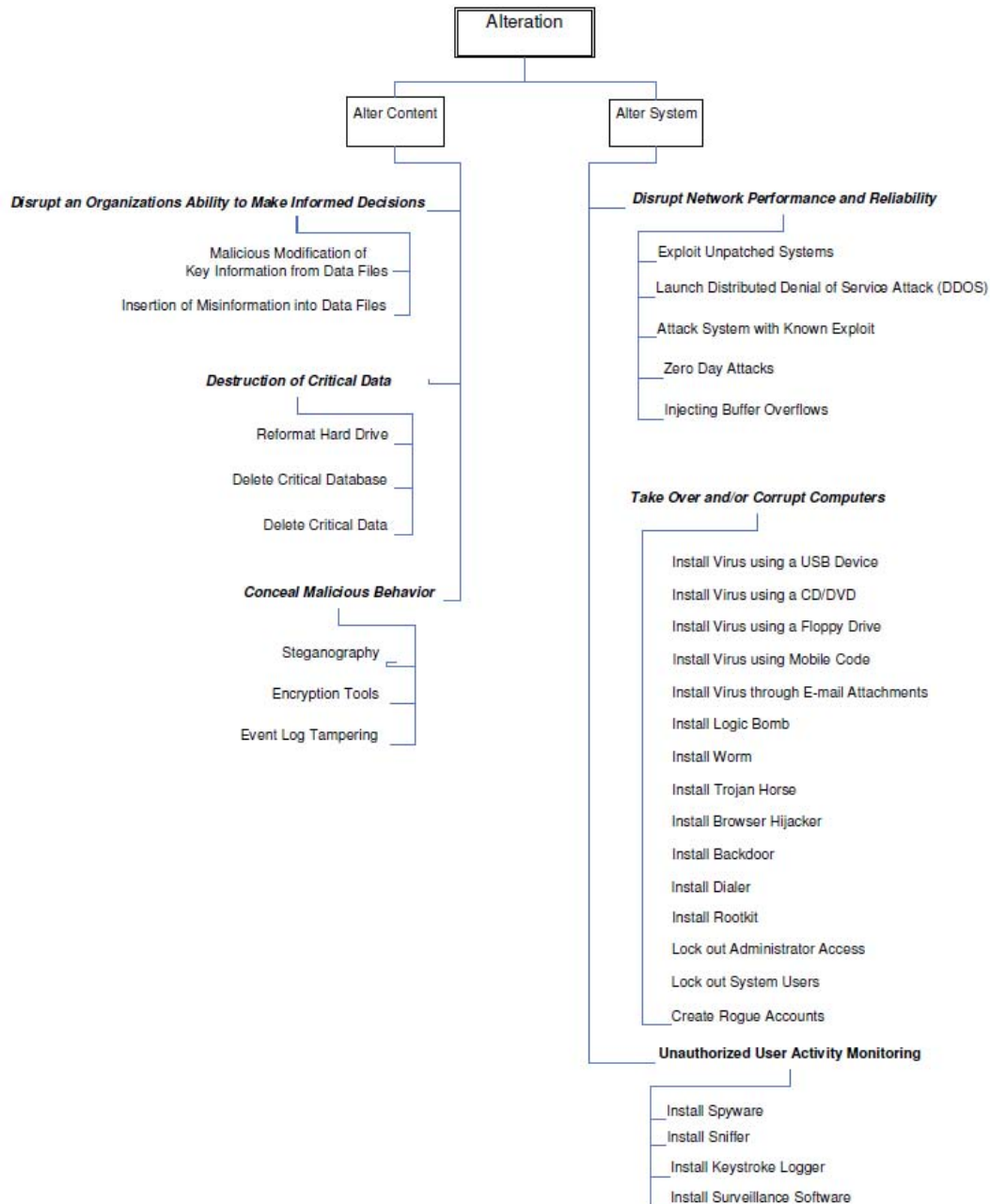


Figure D.1. Insider Threat Decomposed Alteration Model.

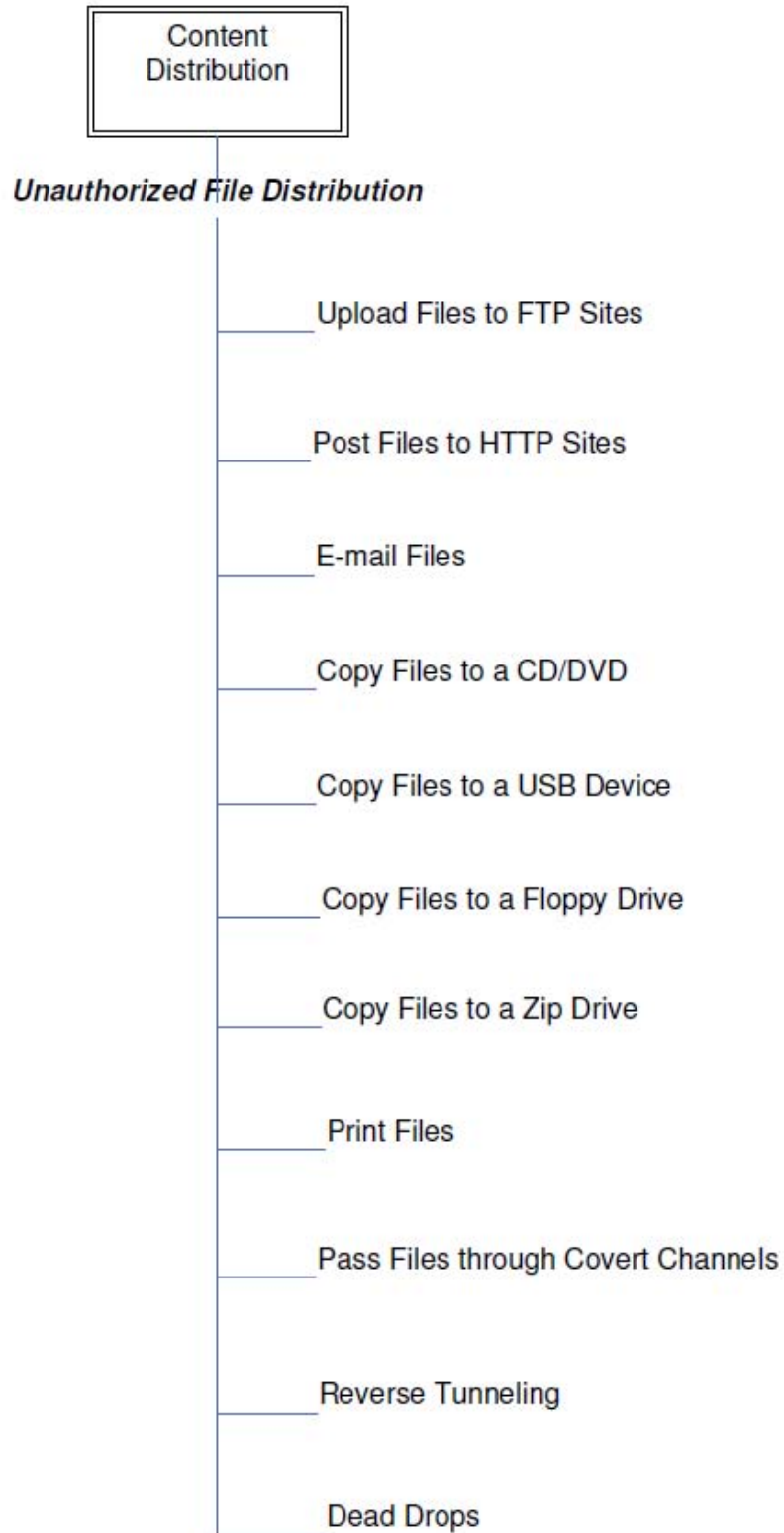


Figure D.2. Insider Threat Distribution Alteration Model.

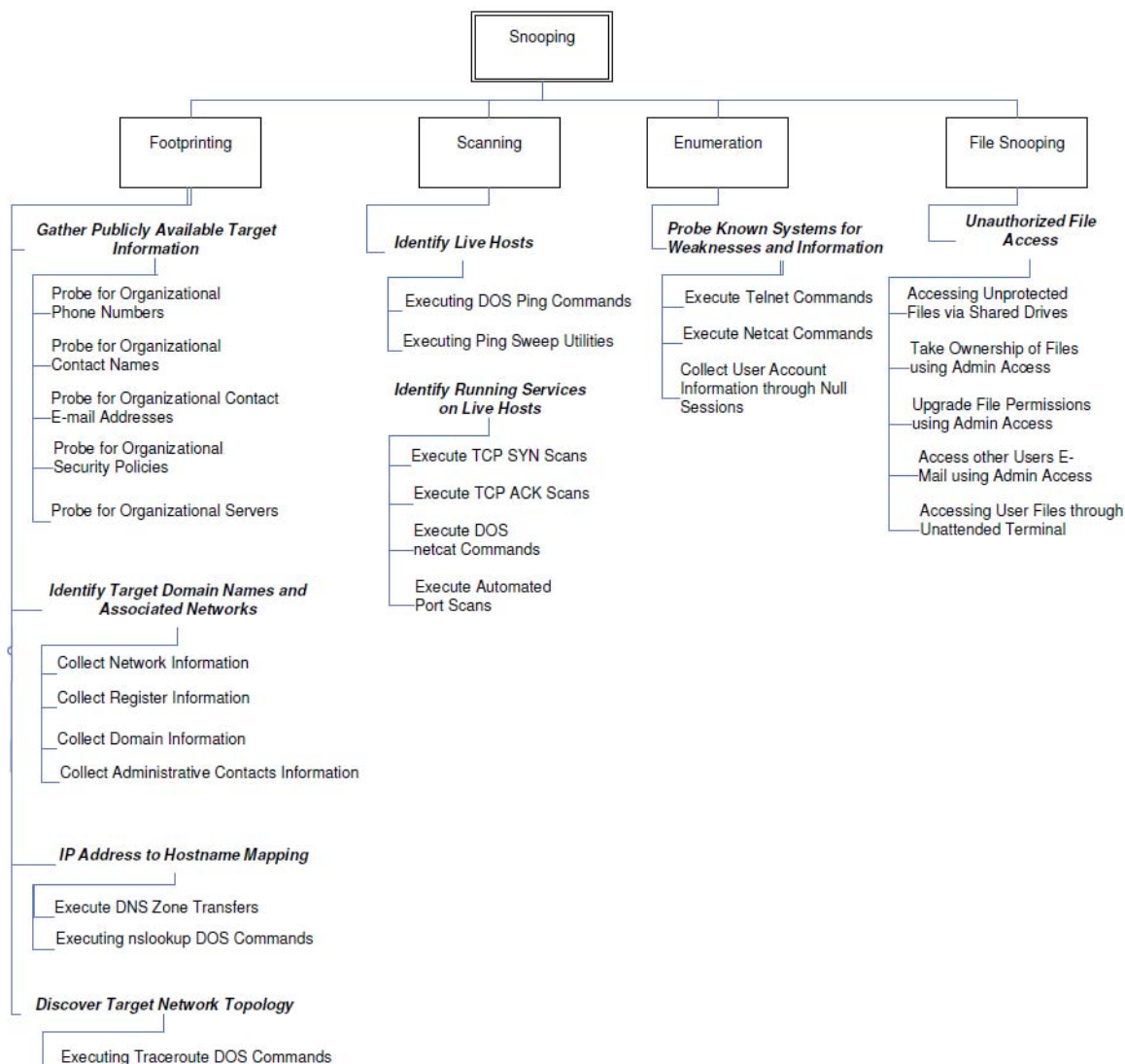


Figure D.3. Insider Threat Distribution Snooping Model.

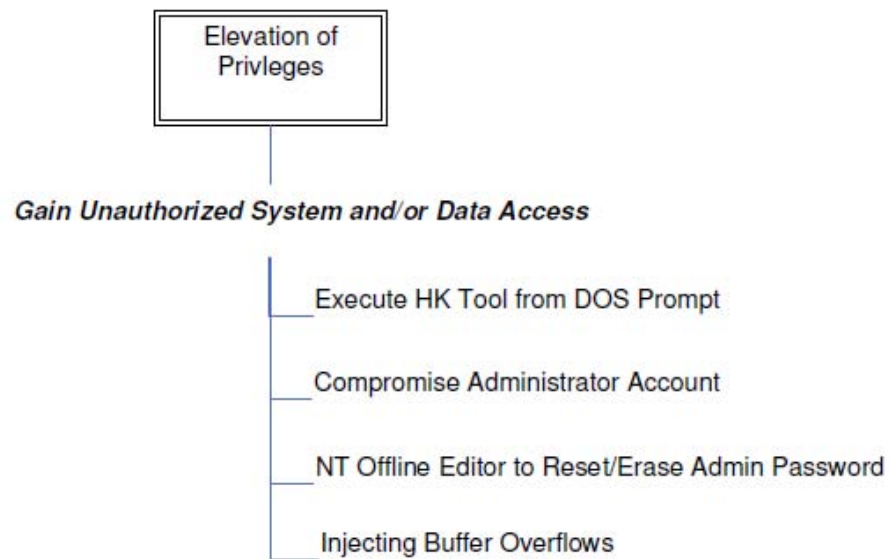


Figure D.4. Insider Threat Distribution Elevation Model.

Appendix E: Insider Threat Use Cases

Scenario 1

A user that possesses the root password uses the *su* command in a terminal window. Use of the *su* command is prohibited by current security policy.

Procedure:

1. User wbai logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “su” on command line
4. User inputs root password
5. User exits root session by using the “exit” command

Results:

Authpriv.info [rsyslog]

```
Feb  7 18:21:01 ubuntu su[2349]: Successful su for root by wbai
Feb  7 18:21:01 ubuntu su[2349]: + /dev/pts/0 wbai:root
Feb  7 18:21:01 ubuntu su[2349]: pam_unix(su:session): session opened
for user root by wbai(uid=1000)
Feb  7 18:21:25 ubuntu su[2349]: pam_unix(su:session): session closed
for user root
```

Audit Log (watching writes to /var/run/utmp)

```
time->Mon Feb  8 19:33:10 2010
type=PATH msg=audit(1265686390.605:992): item=0 name="/var/run/utmp"
inode=2708 dev=00:12 mode=0100664 ouid=0 ogid=43 rdev=00:00
type=CWD msg=audit(1265686390.605:992):  cwd="/home/wbai"
type=SYSCALL msg=audit(1265686390.605:992): arch=40000003 syscall=5
success=yes exit=3 a0=dd0eba a1=88000 a2=f a3=dd0ec0 items=1 ppid=2457
pid=2690 auid=4294967295 uid=1000 gid=1000 euid=0 suid=0 fsuid=0
egid=1000 sgid=1000 fsgid=1000 tty=pts2 ses=4294967295 comm="su"
exe="/bin/su" key="session"
----
time->Mon Feb  8 19:33:12 2010
type=PATH msg=audit(1265686392.446:997): item=0 name="/var/run/utmp"
inode=2708 dev=00:12 mode=0100664 ouid=0 ogid=43 rdev=00:00
type=CWD msg=audit(1265686392.446:997):  cwd="/home/wbai"
type=SYSCALL msg=audit(1265686392.446:997): arch=40000003 syscall=5
success=yes exit=3 a0=dd0eba a1=88000 a2=1 a3=dd0ec0 items=1 ppid=2457
pid=2690 auid=4294967295 uid=1000 gid=1000 euid=0 suid=0 fsuid=0
egid=1000 sgid=1000 fsgid=1000 tty=pts2 ses=4294967295 comm="su"
exe="/bin/su" key="session"
```

Analysis:

Both the system logging and auditing system captured this scenario. The system logs show the successful event of the “su” command and the opening and closing of the root session. With the auditing events the success of the “su” command is inferred by the opening and closing events registered of the root session. While the results for the system logging events are obvious, some interpretation of the audit log is required.

The audit log record shows a session was opened by watching for writes to the */var/run/utmp* database. The *success* field shows the command was successfully executed, the *exe* field shows the resolved pathname for the command, and the *uid* identifies the user executing the command. It is inferred the second audit entry is the closing of the session as the *pid* field has the same value for both entries.

An audit file watch rule to monitor the execution of the */bin/su* file could also monitor for this scenario.

Scenario 2

A user that does not possess the root password uses the “su” command in a terminal window. Use of the “su” command is prohibited by current security policy.

Procedure:

1. User wbai logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “su” on command line
4. User inputs a false root password.

Results:

Authpriv.err [rsyslog]

```
Feb  7 16:40:31 ubuntu su[2459]: pam_authenticate: Authentication failure
```

Authpriv.notice [rsyslog]

```
Feb  7 19:09:07 ubuntu su[2514]: pam_unix(su:auth): authentication failure; logname=wbai uid=1000 euid=0 tty=/dev/pts/2 ruser=wbai rhost=user=root
Feb  7 19:09:08 ubuntu su[2514]: FAILED su for root by wbai
Feb  7 19:09:08 ubuntu su[2514]: - /dev/pts/2 wbai:root
```

Audit system did not capture any events for this session.

Analysis:

The system log captured a pam authentication error at both levels for the authpriv facility, but only the notice level events show a failed attempt to use the “su” command.

The auditing system did not capture failed events as a failed attempt would not open a session causing a write to occur to the */var/run/utmp* database. Again, as noted in the scenario 1 analysis, a file watch on */bin/su* for execution would suffice to capture both successful and unsuccessful attempts. This is only possible since the use of the “su” command is prohibited under all circumstances.

Scenario 3

An authorized user uses the “sudo” command without a valid password in a terminal window. Use of the “sudo” command is allowed for admin personnel listed in */etc/sudoers*.

This scenario would simulate a user leaving the workstation unattended and unsecured, while an insider attempted to exploit the authorized user’s authority without knowing the authorized user’s password.

Procedure:

1. User wbai (authorized admin listed in */etc/sudoers*) logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “sudo -i” (opens a root shell) on command line
4. User inputs invalid user password (3 attempts)

Results:

Authpriv.notice [rsyslog]

```
Feb  7 16:40:31 ubuntu su[2459]: FAILED su for root by wbai
Feb  7 16:40:31 ubuntu su[2459]: - /dev/pts/0 wbai:root
Feb  7 16:40:38 ubuntu su[2461]: pam_unix(su:auth): authentication
failure; logname=wbai uid=1000 euid=0 tty=/dev/pts/0 ruser=wbai rhost=
user=root
Feb  7 16:40:40 ubuntu su[2461]: FAILED su for root by wbai
Feb  7 16:40:40 ubuntu su[2461]: - /dev/pts/0 wbai:root
Feb  7 16:40:47 ubuntu su[2465]: pam_unix(su:auth): authentication
failure; logname=wbai uid=1000 euid=0 tty=/dev/pts/0 ruser=wbai rhost=
user=root
Feb  7 16:40:50 ubuntu su[2465]: FAILED su for root by wbai
Feb  7 16:40:50 ubuntu su[2465]: - /dev/pts/0 wbai:root
Feb  7 16:40:54 ubuntu su[2466]: pam_unix(su:auth): authentication
failure; logname=wbai uid=1000 euid=0 tty=/dev/pts/0 ruser=wbai rhost=
user=root
```

Authpriv.alert [rsyslog]

```
Feb  7 19:10:31 ubuntu sudo:      wbai : 3 incorrect password attempts ;
TTY=pts/2 ; PWD=/home/wbai ; USER=root ; COMMAND=/bin/bash
```

Audit system did not capture any events for this session.

Analysis:

The events recorded at the notice level represent the “sudo” command’s attempt to run a command at the root level. The alert level message properly defines the attempt as a user inputting the “sudo” command without the proper password, but will only register after three attempts are made. The notice level messages can be interpreted as unsuccessful attempts at using the “su” or “sudo” commands. Since the notice level messages do not delineate which command is used, suspicion of insider attacks would be warranted at multiple attempts to use either command unsuccessfully. Inferring from the system log messages monitoring the */bin/su* file in scenarios 1 and 2 would not be recommended as the command is used for the “sudo” command as well.

Scenario 4

Unauthorized user attempts to use “sudo” command with a valid user password.

Procedure:

1. User user1 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “sudo -i” (opens a root shell) on command line
4. User inputs valid user password

Results:

Authpriv.alert

```
Feb  7 19:12:12 ubuntu sudo:    user1 : user NOT in sudoers ; TTY=pts/2  
; PWD=/home/wbai ; USER=root ; COMMAND=/usr/bin/xterm
```

Audit system did not capture any events for this session.

Analysis:

This result is straightforward in its analysis.

Note: All results in scenarios 5 to 24 did not register any events in the system logs. All recorded events were generated by the auditing subsystem in the audit log.

Scenario 5

Admin or root attempts to access user1 directory and file1. File1 in user1 home directory has permission setting "600" (owner read/write only).

Procedure:

1. User wbai (authorized admin) logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types "cd ../user1" (relative pathname) on command line
4. User types "ls ../user1" (relative pathname) on command line
5. User types "cat ../user1/file1" (relative pathname) on command line
6. User types "su" and inputs root password
7. User types "cd ../user1" (relative pathname) on command line
8. User types "ls ../user1" (relative pathname) on command line
9. User types "cat ../user1/file1" (relative pathname) on command line
10. User exits root shell

Results:

User wbai "ls" on "/home/user1"

```
time->Mon Feb  8 13:47:52 2010
type=PATH msg=audit(1265665672.574:1599): item=0 name="../user1"
inode=146870 dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1265665672.574:1599): cwd="/home/wbai"
type=SYSCALL msg=audit(1265665672.574:1599): arch=40000003 syscall=5
success=no exit=-13 a0=9518e30 a1=98800 a2=b788b694 a3=9518e18 items=1
ppid=2798 pid=2987 auid=4294967295 uid=1000 gid=1000 euid=1000
suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts2
ses=4294967295 comm="ls" exe="/bin/ls" key="user1"
```

User wbai "cat" on "/home/user1/file1"

Permission denied. No events generated.

Root "ls" on "/home/user1"

```
time->Mon Feb  8 13:48:35 2010
type=PATH msg=audit(1265665715.422:1621): item=0 name="." inode=146870
dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1265665715.422:1621): cwd="/home/user1"
type=SYSCALL msg=audit(1265665715.422:1621): arch=40000003 syscall=5
success=yes exit=3 a0=9e68dd8 a1=98800 a2=b77e3694 a3=9e6ce18 items=1
ppid=3001 pid=3013 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0
egid=0 sgid=0 fsgid=0 tty=pts2 ses=4294967295 comm="ls" exe="/bin/ls"
key="user1"
```

Root "cat" on "/home/user1/file1"

```
time->Mon Feb  8 19:25:58 2010
type=PATH msg=audit(1265685958.665:839): item=0 name="../user1/file1"
inode=133713 dev=08:01 mode=0100600 ouid=1001 ogid=1001 rdev=00:00
```

```
type=CWD msg=audit(1265685958.665:839):  cwd="/home/user3"  
type=SYSCALL msg=audit(1265685958.665:839): arch=400000003 syscall=5  
success=yes exit=3 a0=bf476ce a1=8000 a2=0 a3=bf4580c items=1  
ppid=2584 pid=2609 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0  
egid=0 sgid=0 fsgid=0 tty=pts2 ses=4294967295 comm="cat" exe="/bin/cat"  
key="user1"
```

Analysis:

For either wbai or root, the “cd” command did not generate an event, but as soon as a command entered requiring reading the directory contents (ie. “ls” command) an event was recorded. User wbai (admin privileges) was not allowed to access user1 directory due to permissions, but root was allowed. Both the successful and unsuccessful attempts to access the directory were recorded. No events were recorded for user wbai’s attempts to access the file. This is due to the fact that the rule monitoring the file never activated as user wbai could not access the directory, much less the file. Root accessing the file was successfully monitored by the auditing system.

Scenario 6

Admin or root attempts to modify system logging (*/etc/rsyslog.conf*) and audit rule (*/etc/audit/audit.rules*) sets.

Procedure:

1. User wbai (authorized admin) logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “vi /etc/rsyslog.conf” (system logging configuration) on command line
4. User inserts text and writes with a force quit
5. User types “vi /etc/audit/audit.rules” (auditing rule set) on command line
6. User inserts text and writes with a force quit
7. User types “su” and inputs root password
8. User types “vi /etc/rsyslog.conf” (system logging configuration) on command line
9. User inserts text and writes with a force quit
10. User types “vi /etc/audit/audit.rules” (auditing rule set) on command line
11. User inserts text and writes with a force quit
12. User exits root shell

Results:

User wbai “vi” on “/etc/rsyslog.conf”

```
time->Mon Feb 22 01:47:09 2010
type=PATH msg=audit(1266832029.629:969): item=0
name="/etc/rsyslog.conf" inode=88947 dev=08:01 mode=0100644 ouid=0
ogid=0 rdev=00:00
type=CWD msg=audit(1266832029.629:969): cwd="/home/wbai"
type=SYSCALL msg=audit(1266832029.629:969): arch=40000003 syscall=85
success=no exit=-22 a0=bfd0e6c a1=bfd0e6c a2=fff a3=1 items=1
ppid=2336 pid=2587 auid=4294967295 uid=1000 gid=1000 euid=1000
suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0
ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="syslog"
```

User wbai “vi” on “/etc/audit/audit.rules”

Permission denied. No events generated.

Root “vi” on “/etc/rsyslog.conf”

```
time->Mon Feb 22 01:34:44 2010
type=PATH msg=audit(1266831284.641:481): item=0
name="/etc/rsyslog.conf" inode=88947 dev=08:01 mode=0100644 ouid=0
ogid=0 rdev=00:00
type=CWD msg=audit(1266831284.641:481): cwd="/home/wbai"
type=SYSCALL msg=audit(1266831284.641:481): arch=40000003 syscall=5
success=yes exit=3 a0=843aa90 a1=0 a2=0 a3=1 items=1 ppid=2416 pid=2430
auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts0 ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="syslog"
```

Root “vi” on “/etc/audit/audit.rules”

```
time->Mon Feb 22 01:35:18 2010
```

```
type=PATH msg=audit(1266831318.189:502): item=0
name="/etc/audit/audit.rules" inode=133782 dev=08:01 mode=0100640
ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1266831318.189:502): cwd="/home/wbai"
type=SYSCALL msg=audit(1266831318.189:502): arch=400000003 syscall=5
success=yes exit=3 a0=8837a98 a1=0 a2=0 a3=1 items=1 ppid=2416 pid=2435
auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts0 ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="audit"
```

Analysis:

User wbai has read access to */etc/rsyslog.conf* but no write access. Any attempt by wbai to write to */etc/rsyslog.conf* will result in an unsuccessful attempt recorded. User wbai had no access to */etc/audit/audit.rules* and did not generate any events. Root had access and the auditing system recorded successful write attempts to both files. Actual or attempted modifications made to the files are unknown. If modifications are successful, a sound backup copy is needed to compare to the active configurations.

Scenario 7

Admin or root attempts to modify system log and audit log

Procedure:

1. User wbai (authorized admin) logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “vi /var/log/syslog.all.log” (system log) on command line
4. User inserts text and writes with a force quit
5. User types “vi /var/log/audit/audit.log” (audit log) on command line
6. User inserts text and writes with a force quit
7. User types “su” and inputs root password
8. User types “vi /var/log/syslog.all.log” (system log) on command line
9. User inserts text and writes with a force quit
10. User types “vi /var/log/audit/audit.log” (audit log) on command line
11. User inserts text and writes with a force quit
12. User exits root shell

Results:

User wbai “vi” on “/var/log/syslog.all.log”

```
time->Mon Feb 22 01:47:09 2010
type=PATH msg=audit(1266832029.629:969): item=0
name="/etc/rsyslog.conf" inode=88947 dev=08:01 mode=0100644 ouid=0
ogid=0 rdev=00:00
type=CWD msg=audit(1266832029.629:969): cwd="/home/wbai"
type=SYSCALL msg=audit(1266832029.629:969): arch=40000003 syscall=85
success=no exit=-22 a0=bfda0e6c a1=bfdale6c a2=fff a3=1 items=1
ppid=2336 pid=2587 auid=4294967295 uid=1000 gid=1000 euid=1000
suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0
ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="syslog"
```

User wbai “vi” on “/var/log/audit/audit.log”

Permission denied. No events generated.

Root “vi” on “/var/log/syslog.all.log”

```
type=PATH msg=audit(1265665818.196:1638): item=0 name="/var/log/"
inode=120515 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1265665818.196:1638): cwd="/home"
type=SYSCALL msg=audit(1265665818.196:1638): arch=40000003 syscall=5
success=yes exit=5 a0=9915128 a1=c1 a2=1a0 a3=9915128 items=2 ppid=3001
pid=3021 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=pts2 ses=4294967295 comm="vi"
exe="/usr/bin/vim.tiny" key="syslog"
```

Root “vi” on “/var/log/audit/audit.log”

```
time->Mon Feb 8 13:50:41 2010
type=PATH msg=audit(1265665841.769:1655): item=0
name="/var/log/audit/audit.log" inode=130093 dev=08:01 mode=0100600
ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1265665841.769:1655): cwd="/home"
```

```
type=SYSCALL msg=audit(1265665841.769:1655): arch=40000003 syscall=5
success=yes exit=3 a0=9950aa0 a1=0 a2=0 a3=1 items=1 ppid=3001 pid=3027
auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts2 ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="audit"
```

Analysis:

Again, user wbai was allowed to read the system log file, but does not have access to the audit log. If a forced write is executed on the system log an audit event is generated, but is unsuccessful. Root actions to modify both logs are recorded, although what modifications are made are unknown without a backup copy for comparison.

Scenario 8

Admin or root attempts to clear system log and audit log.

Procedure:

1. User wbai (authorized admin) logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “rm /var/log/syslog.all.log” (system log) on command line
4. Examine logs
5. User types “rm /var/log/audit/audit.log” (audit log) on command line
6. Examine logs
7. User types “su” and inputs root password
8. User types “rm /var/log/syslog.all.log” (system log) on command line
9. Examine logs
10. User types “rm /var/log/audit/audit.log” (audit log) on command line
11. Examine logs
12. User exits root shell

Results:

User wbai “rm” on “/var/log/syslog.all.log”

```
time->Mon Feb 22 02:32:21 2010
type=PATH msg=audit(1266834741.737:1300): item=1
name="/var/log/syslog.all.log" inode=1654 dev=08:01 mode=0100640
ouid=101 ogid=4 rdev=00:00
type=PATH msg=audit(1266834741.737:1300): item=0 name="/var/log/"
inode=120515 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1266834741.737:1300): cwd="/home/wbai"
type=SYSCALL msg=audit(1266834741.737:1300): arch=40000003 syscall=301
success=no exit=-13 a0=ffffff9c a1=bff396e9 a2=0 a3=bff396e9 items=2
ppid=2336 pid=2794 auid=4294967295 uid=1000 gid=1000 euid=1000
suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0
ses=4294967295 comm="rm" exe="/bin/rm" key="delete"
```

User wbai “rm” on “/var/log/audit/audit.log”

Permission denied. No events generated.

Root “rm” on “/var/log/syslog.all.log”

```
time->Mon Feb 22 02:38:10 2010
type=PATH msg=audit(1266835090.417:1406): item=1
name="/var/log/syslog.all.log" inode=1654 dev=08:01 mode=0100640
ouid=101 ogid=4 rdev=00:00
type=PATH msg=audit(1266835090.417:1406): item=0 name="/var/log/"
inode=120515 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1266835090.417:1406): cwd="/home/wbai"
type=SYSCALL msg=audit(1266835090.417:1406): arch=40000003 syscall=301
success=yes exit=0 a0=ffffff9c a1=bf9ca9d5 a2=0 a3=bf9ca9d5 items=2
ppid=2336 pid=2895 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0
egid=0 sgid=0 fsgid=0 tty=pts0 ses=4294967295 comm="rm" exe="/bin/rm"
key="delete"
----
```

```
time->Mon Feb 22 02:36:29 2010
```

```
type=CONFIG_CHANGE msg=audit(1266834989.794:1384): auid=4294967295  
ses=4294967295 op="add rule" key="syslog" list=4 res=1
```

Root “rm” on “/var/log/audit/audit.log”

Command successful, no audit.log file in directory.

Analysis:

User wbai did not have write access to /var/log/syslog.all.log and generated an unsuccessful event, while failing to generate any event in the attempt to delete /var/log/audit/audit.log due to lack of permissions to the file.

Root removed the syslog file and generated 2 related events in the audit log. The first event watched for any execution of the system calls related to deletion and the other event generated when a change to the existing audit rule monitoring the syslog file was changed due to the change in value of the inode. While root did delete the audit log, no event was generated. For both log deletions by root, the respective log files were not automatically created and both service daemons needed to be restarted to function properly.

The gap in system log and/or audit log time coverage would be an indicator for suspicious activity.

Scenario 9

User attempts to read system log and audit log.

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “cat /var/log/syslog.all.log” (system log) on command line
4. User types “cat /var/log/audit/audit.log” (audit log) on command line
5. User exits shell

Results:

User2 permission denied no record of attempts for both log files.

Analysis:

Due to user2’s lack of authority, unsuccessful attempts to read these files will not be recorded, but as was proven with root access reads, successful attempts will be recorded.

Scenario 10

User attempts to modify system log and audit log.

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “vi /var/log/syslog.all.log” (system log) on command line
4. If step 3 successful, user inserts text and writes with a force quit
5. User types “vi /var/log/audit/audit.log” (audit log) on command line
6. If step 5 successful, user inserts text and writes with a force quit
7. User exits shell

Results:

User2 has permission denied, no record of attempts for both files.

Analysis:

Due to user2’s lack of authority, unsuccessful attempts to modify these files will not be recorded, but as was proven with root access modifies, successful attempts will be recorded.

Scenario 11

User attempts to clear system log and audit log.

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “rm /var/log/syslog.all.log” (system log) on command line
4. Examine logs
5. User types “rm /var/log/audit/audit.log” (audit log) on command line
6. Examine logs
7. User exits shell

Results:

User2 “rm” on “/var/log/syslog.all.log”

```
time->Fri Feb 12 10:18:09 2010
type=PATH msg=audit(1265998689.002:990): item=1
name="/var/log/syslog.all.log" inode=1654 dev=08:01 mode=0100640
ouid=101 ogid=4 rdev=00:00
type=PATH msg=audit(1265998689.002:990): item=0 name="/var/log/"
inode=120515 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1265998689.002:990): cwd="/home/wbai"
type=SYSCALL msg=audit(1265998689.002:990): arch=40000003 syscall=301
success=no exit=-13 a0=ffffff9c a1=bffe06d1 a2=0 a3=bffe06d1 items=2
ppid=2418 pid=2601 auid=4294967295 uid=1002 gid=1002 euid=1002
suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2
ses=4294967295 comm="rm" exe="/bin/rm" key="delete"
```

User2 “rm” on “/var/log/audit/audit.log”

User2 has permission denied, no record of attempt.

Analysis:

Events were generated for user2 attempts to delete the system log file as users have permission to the /var/log directory, but lack write permissions to the log files. Events for deletion attempts of the /var/log/audit/audit.log failed to generate due to lack of permissions to the /var/log/audit directory.

Scenario 12

User attempts to stop/disable rsyslog and/or auditing daemon

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User type “ps -ef” and notes process IDs for both *rsyslogd* and *auditd*
4. User types “kill [rsyslogd pid]” (system log daemon) on command line
5. User types “kill [auditd pid]” (audit log daemon) on command line
6. User exits shell

Results:

User2 “kill” on *rsyslogd*

```
time->Fri Feb 12 10:29:51 2010
type=OBJ_PID msg=audit(1265999391.751:1617): opid=650 oaudit=-1 ouid=101
oses=-1 obj=(none) ocomm="rsyslogd"
type=SYSCALL msg=audit(1265999391.751:1617): arch=40000003 syscall=37
success=no exit=-1 a0=28a a1=f a2=28a a3=0 items=0 ppid=2409 pid=2418
auid=4294967295 uid=1002 gid=1002 euid=1002 suid=1002 fsuid=1002
egid=1002 sgid=1002 fsgid=1002 tty=pts2 ses=4294967295 comm="bash"
exe="/bin/bash" key="kill"
```

User2 “kill” on *auditd*

```
time->Fri Feb 12 10:28:10 2010
type=OBJ_PID msg=audit(1265999290.203:1599): opid=2768 oaudit=-1 ouid=0
oses=-1 obj=(none) ocomm="auditd"
type=SYSCALL msg=audit(1265999290.203:1599): arch=40000003 syscall=37
success=no exit=-1 a0=ad0 a1=f a2=ad0 a3=0 items=0 ppid=2409 pid=2418
auid=4294967295 uid=1002 gid=1002 euid=1002 suid=1002 fsuid=1002
egid=1002 sgid=1002 fsgid=1002 tty=pts2 ses=4294967295 comm="bash"
exe="/bin/bash" key="kill"
```

Analysis:

Events were generated due to a system call watch on the “kill” command. This will generate events for all “kill” commands occurring on the system. As a “kill” command would normally be executed by an administrator, any attempts by normal users would be suspicious. If for example all normal users have *uid* \geq 2000, and all admins have 1000 \geq *uid* $<$ 2000, then a filter on the system call may be devised to only capture “kill” command attempts by normal users.

Scenario 13

User attempts to modify system log and audit configuration

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “vi /etc/rsyslog.conf” (system logging configuraton) on command line
4. If step 3 successful, user inserts text and writes with a force quit
5. User types “vi /etc/audit/audit.rules” (auditing rule set) on command line
6. If step 3 successful, user inserts text and writes with a force quit
7. User exits shell

Results:

User2 “vi” on “/etc/rsyslog.conf”

```
time->Fri Feb 12 10:37:03 2010
type=PATH msg=audit(1265999823.405:1676): item=0
name="/etc/rsyslog.conf" inode=88947 dev=08:01 mode=0100644 ouid=0
ogid=0 rdev=00:00
type=CWD msg=audit(1265999823.405:1676): cwd="/home/wbai"
type=SYSCALL msg=audit(1265999823.405:1676): arch=40000003 syscall=5
success=yes exit=3 a0=9fedab0 a1=0 a2=0 a3=1 items=1 ppid=2418 pid=2843
auid=4294967295 uid=1002 gid=1002 euid=1002 suid=1002 fsuid=1002
egid=1002 sgid=1002 fsgid=1002 tty=pts2 ses=4294967295 comm="vi"
exe="/usr/bin/vim.tiny" key="syslog"
----
time->Fri Feb 12 10:37:10 2010
type=PATH msg=audit(1265999830.824:1680): item=0
name="/etc/rsyslog.conf" inode=88947 dev=08:01 mode=0100644 ouid=0
ogid=0 rdev=00:00
type=CWD msg=audit(1265999830.824:1680): cwd="/home/wbai"
type=SYSCALL msg=audit(1265999830.824:1680): arch=40000003 syscall=85
success=no exit=-22 a0=bf8d2c3c a1=bf8d3c3c a2=fff a3=1 items=1
ppid=2418 pid=2843 auid=4294967295 uid=1002 gid=1002 euid=1002
suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2
ses=4294967295 comm="vi" exe="/usr/bin/vim.tiny" key="syslog"
```

User2 “vi” on “/etc/audit/audit.rules”

User has permission denied, no record of attempts for audit.rules

Analysis:

Users have read access to the syslog configuration file, but no write access. A successful “vi” event is recorded as the program reads the file and an unsuccessful “vi” event is recorded due to an attempt to write to the file.

Again, no events are recorded for the audit rule set as the user lacks permissions for the */etc/audit* directory.

Scenario 14

User attempts to add user to system

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “adduser” on command line
4. User exits shell

Results:

User has permission denied, no record of attempts

Analysis:

User attempts to add users via the GUI will fail as the necessary options are grayed out due to the lack of permission of the normal user. However attempts may still be made via the command line. However the user still lacks permissions as only admin group or root may add users to the system via command line.

Scenario 15

User attempts to modify user/group information

(*/etc/group, /etc/passwd, /etc/gshadow, /etc/shadow, /etc/sudoers, /etc/security/opasswd*)

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “vi [filename]” on command line
4. User exits shell

Results:

User has permission denied, no record of attempts for all files except */etc/passwd* and */etc/group*

User2 attempts to modify */etc/group* with forced write

```
time->Fri Feb 12 11:12:44 2010
type=PATH msg=audit(1266001964.526:1925): item=0 name="/etc/group"
inode=88990 dev=08:01 mode=0100644 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1266001964.526:1925): cwd="/home/wbai"
type=SYSCALL msg=audit(1266001964.526:1925): arch=40000003 syscall=5
success=no exit=-13 a0=9554a18 a1=241 a2=1a4 a3=81a4 items=1 ppid=2418
pid=3011 auid=4294967295 uid=1002 gid=1002 euid=1002 suid=1002
fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2 ses=4294967295
comm="vi" exe="/usr/bin/vim.tiny" key="identity"
```

Analysis:

User2 had access to read both */etc/passwd* and */etc/group*, but no write permission. Audit would record event if a forced write was attempted (“:w!”). Other files were not recorded and not allowed access.

Scenario 16

User A attempts to access User B directory

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “ls ../user1” (relative pathname) on command line
4. User exits shell

Results:

User user2 "ls" on "/home/user1"

```
time->Fri Feb 12 10:10:56 2010
type=PATH msg=audit(1265998256.018:764): item=0 name="../user1"
inode=146870 dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1265998256.018:764): cwd="/home/wbai"
type=SYSCALL msg=audit(1265998256.018:764): arch=40000003 syscall=5
success=no exit=-13 a0=8df9e30 a1=98800 a2=b784c694 a3=8df9e18 items=1
ppid=2418 pid=2443 auid=4294967295 uid=1002 gid=1002 euid=1002
suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2
ses=4294967295 comm="ls" exe="/bin/ls" key="user1"
```

Analysis:

The unsuccessful attempt to access User1 directory generates an event noting the *uid* of the account attempting the directory read. Such unsuccessful attempts to access restricted directories may indicate suspicious behavior.

Scenario 17

User A attempts to read User B file without group access or file access.

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “cat ../user1/file1” (relative filename) on command line
4. User exits shell

Results:

Permission denied, no events are generated.

Analysis:

As with similar attempts made by admin users in scenario 5, no events are generated due to lack of access to the directory in which the file resides.

Scenario 18

Unauthorized user attempts to read a restricted file (group access, no file access)

Procedure:

1. User1 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “cd ../user3” (relative filename) on command line
4. User types “cat file3” on command line
5. User exits shell

Results:

User1 “cat” on “file3”

```
time->Fri Feb 12 14:55:26 2010
type=PATH msg=audit(1266015326.818:598): item=0 name="file3"
inode=133714 dev=08:01 mode=0100600 ouid=1003 ogid=1001 rdev=00:00
type=CWD msg=audit(1266015326.818:598): cwd="/home/user3"
type=SYSCALL msg=audit(1266015326.818:598): arch=40000003 syscall=5
success=no exit=-13 a0=bfe626cf a1=8000 a2=0 a3=bfe6221c items=1
ppid=2316 pid=2347 auid=4294967295 uid=1001 gid=1001 euid=1001
suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001 tty=pts0
ses=4294967295 comm="cat" exe="/bin/cat" key="user3"
```

Analysis:

Unsuccessful attempts to read restricted files are recorded. Again, numerous unsuccessful attempts may indicate suspicious behavior and allow targeted monitoring of users attempting such actions.

Scenario 19

User2 attempts to modify user1 directory permissions.

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “chmod 666 ../user1” (relative filename) on command line
4. User types “chown user2 ../user1” (relative filename) on command line
5. User types “chgrp group2 ../user1” (relative filename) on command line
6. User exits shell

Results:

User2 “chmod” on “/home/user1”

```
time->Fri Feb 12 11:18:30 2010
type=PATH msg=audit(1266002310.183:1974): item=0 name="/home/user1"
inode=146870 dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1266002310.183:1974): cwd="/home/wbai"
type=SYSCALL msg=audit(1266002310.183:1974): arch=40000003 syscall=306
success=no exit=-1 a0=ffffff9c a1=87d6130 a2=1ff a3=87d6130 items=1
ppid=2418 pid=3056 auid=4294967295 uid=1002 gid=1002 euid=1002
suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2
ses=4294967295 comm="chmod" exe="/bin/chmod" key="perm_mod"
```

User2 “chown” on “/home/user1”

```
time->Fri Feb 12 11:18:49 2010
type=PATH msg=audit(1266002329.786:1981): item=0 name="/home/user1"
inode=146870 dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1266002329.786:1981): cwd="/home/wbai"
type=SYSCALL msg=audit(1266002329.786:1981): arch=40000003 syscall=298
success=no exit=-1 a0=ffffff9c a1=969d8e0 a2=3ea a3=ffffff items=1
ppid=2418 pid=3063 auid=4294967295 uid=1002 gid=1002 euid=1002
suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2
ses=4294967295 comm="chown" exe="/bin/chown" key="perm_mod"
```

User2 “chgrp” on “/home/user1”

```
time->Fri Feb 12 11:19:02 2010
type=PATH msg=audit(1266002342.050:1984): item=0 name="/home/user1"
inode=146870 dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1266002342.050:1984): cwd="/home/wbai"
type=SYSCALL msg=audit(1266002342.050:1984): arch=40000003 syscall=298
success=no exit=-1 a0=ffffff9c a1=90a08d0 a2=ffffff a3=3ea items=1
ppid=2418 pid=3064 auid=4294967295 uid=1002 gid=1002 euid=1002
suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts2
ses=4294967295 comm="chgrp" exe="/bin/chgrp" key="perm_mod"
```

Analysis:

Any attempt to change attributes of a restricted directory is recorded. However, due to the limitation of the auditing subsystem, file watch rules will record both successful and unsuccessful attempts of reads, writes, executes, or attribute changes or any combination

of these rights. As such attribute changes should not be a common occurrence and may indicate suspicious behavior.

Scenario 20

User2 attempts to modify *file1* permissions without group access or file access.

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “chmod 666 ../user1/file1” (relative filename) on command line
4. User types “chown user2 ../user1/file1” (relative filename) on command line
5. User types “chgrp group2 ../user1/file1” (relative filename) on command line
6. User exits shell

Results:

User has permission denied, no record of attempts.

Analysis:

As with similar attempts made by admin users in scenario 5, no events are generated due to lack of access to the directory in which the file resides.

Scenario 21

User1 attempts to modify *file3* permissions (group access and no file access)

Procedure:

1. User1 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “cd ../user3” (relative pathname)
4. User types “chmod 666 file3” on command line
5. User types “chown user1 file3” on command line
6. User exits shell

Results:

User1 “chmod” on “file3”

```
time->Fri Feb 12 14:55:37 2010
type=PATH msg=audit(1266015337.096:609): item=0 name="file3"
inode=133714 dev=08:01 mode=0100600 ouid=1003 ogid=1001 rdev=00:00
type=CWD msg=audit(1266015337.096:609): cwd="/home/user3"
type=SYSCALL msg=audit(1266015337.096:609): arch=40000003 syscall=306
success=no exit=-1 a0=ffffff9c a1=9c5f130 a2=1b0 a3=9c5f130 items=1
ppid=2316 pid=2357 auid=4294967295 uid=1001 gid=1001 euid=1001
suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001 tty=pts0
ses=4294967295 comm="chmod" exe="/bin/chmod" key="perm_mod"
```

User1 “chown” on “file3”

```
time->Fri Feb 12 14:55:50 2010
type=PATH msg=audit(1266015350.905:615): item=0 name="file3"
inode=133714 dev=08:01 mode=0100600 ouid=1003 ogid=1001 rdev=00:00
type=CWD msg=audit(1266015350.905:615): cwd="/home/user3"
type=SYSCALL msg=audit(1266015350.905:615): arch=40000003 syscall=298
success=no exit=-1 a0=ffffff9c a1=8a418e0 a2=3e9 a3=ffffffff items=1
ppid=2316 pid=2363 auid=4294967295 uid=1001 gid=1001 euid=1001
suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001 tty=pts0
ses=4294967295 comm="chown" exe="/bin/chown" key="perm_mod"
```

Analysis:

Unsuccessful attempts to modify restricted file attributes are recorded. Again, numerous unsuccessful attempts may indicate suspicious behavior and allow targeted monitoring of users attempting such actions.

Scenario 22

User attempts to mount external flash media (external media is prohibited by security policy)

Procedure:

1. User2 logs into the system
2. User plugs in a USB Flash Drive
3. User exits shell

Results:

```
time->Mon Feb  8 16:28:18 2010
type=PATH msg=audit(1265675298.444:32): item=1 name=(null) inode=11668
dev=00:0f mode=060660 ouid=0 ogid=6 rdev=08:11
type=PATH msg=audit(1265675298.444:32): item=0 name="/media/DEANNABAI"
inode=146918 dev=08:01 mode=040700 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1265675298.444:32): cwd="/"
type=SYSCALL msg=audit(1265675298.444:32): arch=40000003 syscall=21
success=yes exit=0 a0=8916260 a1=8916270 a2=8916288 a3=c0ed0006 items=2
ppid=1691 pid=2699 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0
egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="mount"
exe="/bin/mount" key="export"
```

Analysis:

Any *automount* operation is performed at the root level. As the introduction of external media is prohibited any event recorded by this auditing rule can be deemed as suspicious behavior.

Scenario 23

User1 clears */home/user1/.bash_history* file.

Procedure:

1. User1 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types “rm .bash_history”
4. User exits shell

Results:

```
time->Mon Feb  8 19:33:29 2010
type=PATH msg=audit(1265686409.665:1030): item=1 name=".bash_history"
inode=132715 dev=08:01 mode=0100600 ouid=1001 ogid=1001 rdev=00:00
type=PATH msg=audit(1265686409.665:1030): item=0 name="/home/user1"
inode=146870 dev=08:01 mode=040700 ouid=1001 ogid=1001 rdev=00:00
type=CWD msg=audit(1265686409.665:1030):  cwd="/home/user1"
type=SYSCALL msg=audit(1265686409.665:1030): arch=40000003 syscall=301
success=yes exit=0 a0=ffffff9c a1=bfald6c9 a2=0 a3=bfald6c9 items=2
ppid=2700 pid=2723 auid=4294967295 uid=1001 gid=1001 euid=1001
suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001 tty=pts2
ses=4294967295 comm="rm" exe="/bin/rm" key="delete"
----
time->Wed Nov 18 09:59:16 2009
type=CONFIG_CHANGE msg=audit(1258567156.321:98): auid=4294967295
ses=4294967295 op="updated rules" path="/home/user1/.bash_history"
key="user1history" list=4 res=1
```

Analysis:

As noted before with the deletion of the system log file by root, 2 related events are generated. The first event watched for any execution of the system calls related to deletion and the other event generated when a change to the existing audit rule monitoring the history file was changed due to the change in value of the inode. The usefulness of monitoring this file is not recreating the commands executed by the user. This is useful within a very limited timeframe depending upon the settings for the history file, but more for the indicator that a user is actively trying to hide their actions.

Scenario 24

Unauthorized user attempts to delete a file with and without group access

Procedure:

1. User2 logs into the system
2. User opens terminal window via drop down menu on GUI desktop
3. User types "rm ../user1/file1"
4. User2 logs out
5. User1 logs into the system
6. User opens terminal window via drop down menu on GUI desktop
7. User types "rm ../user3/file3"
8. User exits shell

Results:

User2 "rm" on "/home/user1/file1"

Permission denied, no events generated.

User1 "rm" on "/home/user3/file3"

```
time->Mon Feb 22 04:08:49 2010
type=PATH msg=audit(1266840529.634:2002): item=1 name="../user3/file3"
inode=133714 dev=08:01 mode=0100600 ouid=1003 ogid=1001 rdev=00:00
type=PATH msg=audit(1266840529.634:2002): item=0 name="../user3/"
inode=146872 dev=08:01 mode=040750 ouid=1003 ogid=1001 rdev=00:00
type=CWD msg=audit(1266840529.634:2002): cwd="/home/wbai"
type=SYSCALL msg=audit(1266840529.634:2002): arch=40000003 syscall=301
success=no exit=-13 a0=ffffff9c a1=bfb896da a2=0 a3=bfb896da items=2
ppid=3511 pid=3531 auid=4294967295 uid=1001 gid=1001 euid=1001
suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001 tty=pts0
ses=4294967295 comm="rm" exe="/bin/rm" key="delete"
```

Analysis:

As with previous scenarios, user2's attempts to delete file1 failed to register events due to its lack of access to the directory. User1's attempts to delete file3 are recorded due to the system call watch on the deletion system calls. This may also be recorded by monitoring write attempts to the file itself as well.

Bibliography

- [1] Fyffe, George. "Addressing the insider threat," *Network Security*, 11 – 14 (March 2008).
- [2] Furnell, Steven. "Enemies within: the problem of insider attacks," *Computer Fraud & Security*, 6 – 11 (July 2004).
- [3] King, William H. *Development of a Malicious Insider Composite Vulnerability Assessment Methodology*. MS Thesis, AFIT/GIA/ENG/06-06. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 2006.
- [4] Butts, Jonathan W. *Formal Mitigation Strategies for the Insider Threat: A Security Model and Risk Analysis Framework*. MS Thesis, AFIT/GIA/ENG/06-02. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2006.
- [5] Levoy, Terry E. *Development of a Methodology for Customizing Insider Threat Auditing on a Microsoft Windows XP® Operating System*. MS Thesis, AFIT/GIA/ENG/06-07. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 2006.
- [6] Skoudis, Ed and Tom Liston. *Counter Hack Reloaded, Second Edition: A Step-by-Step Guide to Computer Attacks and Effective Defenses*. Stoughton MA, Prentice Hall, 2006.
- [7] Department of Commerce. *Standards for Security Categorization of Federal Information and Information Systems*. Federal Information Processing Standards Publication (FIPS PUB) 199. Gaithersburg, MD: National Institute of Standards and Technology, February 2004.
- [8] Pastore, Mike and Emmett Dulaney. *CompTIA Security+ Study Guide: Deluxe Edition*, Indianapolis: Wiley Publishing, Inc., 2006.
- [9] United States District Court for the Eastern District of Virginia, Alexandria Division. *The United States of America v. Robert Phillip Hansen – Affidavit in*

Support of Criminal Complaint, Arrest Warrant and Search Warrants. 25 February 2010

<http://www.fbi.gov/pressrel/pressrel01/affidavit.pdf>.

- [10] Bishop, Matt, Sophie Engle, Sean Peisert, et al. "We have met the enemy and he is us," *New Security Paradigms Workshop 2008 (NSPW'08)*, Lake Tahoe, California. 22-25 September 2008.
- [11] Cappelli, Dawn, Andrew Moore, et al. *Common Sense Guide to Prevention and Detection of Insider Threats 3rd Edition – Version 3.1*, Pittsburgh PA: CERT – Software Engineering Institute, January 2009.
- [12] Anderson, Robert H., Thomas Bozek, et al. "Research on Mitigating the Insider Threat to Information Systems - #2," *Proceeding of a Workshop Held August 2000*, Arlington VA: RAND, August 2000.
- [13] Gabrielson, Bruce, Karen M. Goertzel, et al. *The Insider Threat to Information Systems: A State-of-the-Art Report*. Contract SPO700-98-D-4002. Herndon VA: Information Assurance Technology Analysis Center (IATAC), October, 2008.
- [14] CSO Magazine, US Secret Service, Carnegie Mellon CERT/CC, and Microsoft Corp. "2007 E-Crime Watch Survey," 2007.
- [15] CSO Magazine, US Secret Service, Carnegie Mellon CERT/CC, and Microsoft Corp. "2006 E-Crime Watch Survey," 2006.
- [16] Peters, Sara. "14th Annual CSI Computer Crime and Security Survey Executive Summary," Computer Security Institute, December 2009.
- [17] US Secret Service and Carnegie Mellon University Software Engineering Institute CERT. *Insider Threat Study: Illicit Cyber Activity in the Government Sector*, January 2008.
- [18] National Infrastructure Advisory Council, *Final Report and Recommendations on the Insider Threat to Critical Infrastructures*, April 2008.
- [19] US Secret Service and Carnegie Mellon University Software Engineering Institute CERT. *Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector*, June 2005.

- [20] Department of Defense. *DoD Insider Threat Mitigation, Final Report of the Insider Threat Integrated Process Team*, 24 April 2000.
- [21] Department of Commerce. *Risk Management Guide for Information Technology Systems*. Special Publication (SP) 800-30. Gaithersburg, MD: National Institute of Standards and Technology, July 2002.
- [22] Brackney, Richard C. and Robert H. Anderson. "Understanding the Insider Threat," *Proceedings of a March 2004 Workshop*, Rockville MD: RAND, March 2004.
- [23] Department of Commerce. *Generally Accepted Principles and Practices for Securing Information Technology Systems*. Special Publication (SP) 800-14. Gaithersburg, MD: National Institute of Standards and Technology, September 1996.
- [24] Department of Commerce. *An Introduction to Computer Security: The NIST Handbook*. Special Publication (SP) 800-12. Gaithersburg, MD: National Institute of Standards and Technology, October 1995.
- [25] Department of Commerce. *Minimum Security Requirements for Federal Information and Information Systems*. Federal Information Processing Standards Publication (FIPS PUB) 200. Gaithersburg, MD: National Institute of Standards and Technology, March 2006.
- [26] Moore, Andrew P., Dawn M. Cappelli, et al. "An Experience Using System Dynamics to Facilitate an Insider Threat Workshop." Unpublished Paper, Carnegie Mellon University CERT Software Engineering Institute, 2006.
- [27] Department of Commerce. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Special Publication (SP) 800-94. Gaithersburg, MD: National Institute of Standards and Technology, February 2007.
- [28] Department of Commerce. *Guide to Computer Security Log Management*. Special Publication (SP) 800-92. Gaithersburg, MD: National Institute of Standards and Technology, October 1995.
- [29] Lonvick, C. "The BSD syslog Protocol." Request for Comment (RFC) 3164. Network Working Group. August 2001.

- [30] “Rsyslog – History.” 10 October 2009
<http://www.rsyslog.com/doc-history.html>.
- [31] Hermans, Shawn. “Detect Insider Threats with Linux Auditing.” *Linux.com*. 22 June 2007. 10 October 2009
<http://www.linux.com/archive/feature/114422>.
- [32] Novell, Inc. “SUSE Linux Enterprise 10 SP1: The Linux Audit Framework.” 8 May 2008.
- [33] Department of Defense. *Unix Security Technical Implementation Guide (STIG) Version 5, Release 1*. Unix STIG v5r1. Washington: Defense Information Systems Agency (DISA), 28 March 2006.
- [34] CSO Magazine, US Secret Service, and Carnegie Mellon CERT/CC. “2004 E-Crime Watch Survey Summary of Findings,” 2004.
- [35] CSO Magazine, US Secret Service, and Carnegie Mellon CERT/CC. “2005 E-Crime Watch Survey Summary of Findings,” 2005.
- [36] Power, Richard. “2002 CSI/FBI Computer Crime and Security Survey,” *Computer Security Issues & Trends*, Vol. VIII, No. 1. Computer Security Institute, Spring 2002.
- [37] Richardson, Robert. “8th Annual CSI Computer Crime and Security Survey,” Computer Security Institute, 2003.
- [38] Gordon, Lawrence A., Martin P. Loeb, et al. “9th Annual CSI Computer Crime and Security Survey,” Computer Security Institute, 2004.
- [39] Department of Justice. “2005 FBI Computer Crime Survey,” Federal Bureau of Investigation, 2005.
- [40] Gordon, Lawrence A., Martin P. Loeb, et al. “11th Annual CSI Computer Crime and Security Survey,” Computer Security Institute, 2006.
- [41] Richardson, Robert. “2008 CSI Computer Crime & Security Survey,” Computer Security Institute, 2008.

- [42] Grubb, Steve. Presentation slides, "Audit and IDS." Red Hat Summit, Boston, 18-20 June 2008. 3 November 2009
<http://people.redhat.com/sgrubb/audit/audit-ids.pdf>
- [43] Levoy, Terry, Grimaila, Michael R., and Mills, Robert, "A Methodology for Customizing Security Auditing Templates for Malicious Insider Detection," Proceedings of the 8th International Symposium on System and Information Security (ISIS 2006); Sao Jose dos Campos, Sao Paulo, Brazil; Nov. 08-10, 2006.
- [44] Myers, Justin, Grimaila, Michael R., and Mills, Robert F., "Towards Insider Threat Detection using Web Server Logs," *Proceedings of the Cyber Security and Information Intelligence Research Workshop (CSIIRW 2009)*, Oak Ridge National Laboratory, Oak Ridge, TN, April 13-15, 2009.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 25 - 03 - 2010		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From - To) April 2009 - March 2010	
4. TITLE AND SUBTITLE Development of a Methodology For Customizing Insider Threat Auditing on a Linux Operating System				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) William T. Bai, Master Sergeant (MSgt), USAF				5d. PROJECT NUMBER ENG09-337	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/ENG) 2950 Hobson Way Wright-Patterson AFB, OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCO/ENG/10-01	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Security Agency Attn: Lt Col Joseph L. Wolfkiel 9800 Savage Rd Ste 6767 Ft Meade, MD 20755-6767 Commercial 410-854-5401 DSN 244-5401 j.wolfki@radium.ncsc.mil				10. SPONSOR/MONITOR'S ACRONYM(S) NSA/CND R&T	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Insider threats can pose a great risk to organizations and by their very nature are difficult to protect against. Auditing and system logging are capabilities present in most operating systems and can be used for detecting insider activity. However, current auditing methods are typically applied in a haphazard way, if at all, and are not conducive to contributing to an effective insider threat security policy. This research develops a methodology for designing a customized auditing and logging template for a Linux operating system. An intent-based insider threat risk assessment methodology is presented to create use case scenarios tailored to address an organization's specific security needs and priorities. These organization specific use cases are verified to be detectable via the Linux auditing and logging subsystems and the results are analyzed to create an effective auditing rule set and logging configuration for the detectable use cases. Results indicate that creating a customized auditing rule set and system logging configuration to detect insider threat activity is possible.					
15. SUBJECT TERMS Insider Threat; Auditing; Logging; Linux					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
REPORT	ABSTRACT	c. THIS PAGE			Robert F. Mills, PhD (ENG)
U	U	U	UU	113	19b. TELEPHONE NUMBER (Include area code) (937) 255-6565 ext 4527; email: robert.mills@afit.edu

Standard Form 298 (Rev: 8-98)

Prescribed by ANSI Std. Z39-18